The Dissertation Committee for Alexander Yun-chung Liu
certifies that this is the approved version of the following dissertation:

Active Learning in Cost-Sensitive Environments

Committee:

Joydeep Ghosh, Supervisor

J.K. Aggarwal

Lizy Kurian John

Cheryl Martin

Maytal Saar-Tsechansky

Active Learning in Cost-Sensitive Environments

by

Alexander Yun-chung Liu, B.S., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2009

For Rachel and her generation

Acknowledgments

This section has been rather difficult to write, and to be honest, I have put it off for last. It is difficult to adequately express how thankful I am to all those that I know. I won't go so far to say that this is the "most" difficult part of the dissertation to write (I'll stick with "rather" difficult), but mostly because I'm trying not to accidentally imply that the rest of this was easy to write. Regardless, I am in the sarcastically unfortunate position to be blessed with a very large number of people to thank. I could possibly start with "If I were to list all the people who have helped me over the years, then this section would eclipse the rest of this dissertation in size," which, while true, either implies I have a large list of people to thank or that the rest of this dissertation doesn't have much content. I could also have started with "Words cannot express my gratitude towards all those that have helped me over the years," which, while also true, immediately opens this dissertation to the snide criticism that apparently words also cannot express the technical merit of my work over the years.

I've also come to the realization that if the only way someone knows how much I appreciate them is to have to look it up in some tiny print in some random section of writing, then I'm probably doing something wrong. I know I'm not going to win any awards for my capability to show gratitude, but that's something I'm going to try to change. So for those of you looking for a laundry list of people and comments, that's not what this section is going to be like. Somehow, I'd rather try

to thank you in person.

So, saving a few exceptions, I've decided just to list categories of people. Without further dithering, I would like to thank the following groups of people in no particular order:

Thank you to all the members of my dissertation committee for selflessly giving of their time, their generosity, and their many interesting and constructive comments.

Thank you to all those that have funded my research over the years, including all those affiliated with the Texas Exes Scholarship program, the MCD fellowship, the Thrust fellowship, and various travel fellowships. I would also like to express my appreciation for the generous funding from ARL, Dr. Ghosh, and Dr. Konana over the years.

To the various colleagues I've had the chance to work with over the years, particularly all past and current members of IDEAL and CIADS where this research was conducted: Thank you for all the discussions about research and about what it means to perform research. Thank you for the technical and emotional support. Thank you for being cool people to spend time with. I would especially like to acknowledge all the co-authors I've had the pleasure of working with. I hope you've all had as much fun as I've had collaborating with you. I couldn't have done it alone.

To Dr. Joydeep Ghosh, my supervising professor, and to Dr. Cheryl Martin, who basically co-supervised this dissertation, thank you for the years of guidance and support, both in the research world and otherwise. I've only had a small taste of what it is like to be a mentor, and I have no idea how the two of you make it look so easy. Thank you for your patience, your understanding, and your direction. The amount I've learned by being a member of your respective labs amazes me.

Finally, I would like to thank everyone that's left. That includes all of you

friends and family members that will only ever read this part of the dissertation to see what I said about you. Apologies to those disappointed that this section isn't just a list of people, but please wait until the next we meet so I can thank you properly in person.

Alexander Yun-chung Liu

The University of Texas at Austin
December 2009

Active Learning in Cost-Sensitive Environments

Publication No. _____

Alexander Yun-chung Liu, Ph.D.
The University of Texas at Austin, 2009

Supervisor: Joydeep Ghosh

Abstract

Active learning techniques aim to reduce the amount of labeled data required for a supervised learner to achieve a certain level of performance. This can be very useful in domains where unlabeled data is easy to obtain but labelling data is costly. In this dissertation, I introduce methods of creating computationally efficient active learning techniques that handle different misclassification costs, different evaluation metrics, and different label acquisition costs. This is accomplished in part by developing techniques from utility-based data mining typically not studied in conjunction with active learning.

I first address supervised learning problems where labeled data may be scarce, especially for one particular class. I revisit claims about resampling, a particularly

popular approach to handling imbalanced data, and cost-sensitive learning. The presented research shows that while resampling and cost-sensitive learning can be equivalent in some cases, the two approaches are not identical.

This work on resampling and cost-sensitive learning motivates a need for active learners that can handle different misclassification costs. After presenting a cost-sensitive active learning algorithm, I show that this algorithm can be combined with a proposed framework for analyzing evaluation metrics in order to create an active learning approach that can optimize any evaluation metric that can be expressed as a function of terms in a confusion matrix. Finally, I address methods for active learning in terms of different utility costs incurred when labeling different types of points, particularly when label acquisition costs are spatially driven.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Active Learning

Supervised learning is a standard task in machine learning. However, the drawback of many supervised learning approaches is that they require a large amount of labeled training data in order to perform well. Creating such a large corpus of labeled training data can be extremely costly and time consuming.

There are many domains where labeled data is costly to obtain but unlabeled data is cheap to acquire. In such a setting, active learning can be used to reduce the number of points required for a supervised learner to achieve a certain level of performance. As opposed to random sampling, an active learning approach achieves this by letting the learning algorithm itself indicate which points should be labeled.

In this dissertation, I will present methods for improving current abilities of active learning. In particular, I will introduce a computationally efficient method of handling such things as points with different misclassification costs and points with different labeling costs. To do this, I will leverage techniques in utility-based data mining typically not connected to active learning.

1.2 Utility-based Data Mining

Utility-based data mining combines and unites several previously separate problems in data mining under a single banner, including the task of active learning. All problems contained within the utility-based data mining hierarchy have one common theme that separates them from "normal" data mining: the problems involve unequal costs, benefits, or utilities for performing a certain action. For example, in a typical classification setting, there is an equal misclassification cost for each class. In cost-sensitive learning, a utility-based data mining version of classification, there could be an unequal misclassification cost for each class or even an unequal misclassification cost for each point in the dataset.

Cost-sensitive learning is one of the problems which fits very naturally in the domain of utility-based data mining algorithms. Cost-sensitive learning is a supervised classification problem where misclassifying a data point incurs a certain cost. The goal of cost-sensitive learning is to minimize the total or expected cost of misclassification incurred on some test set. A problem related to cost-sensitive learning is the imbalanced dataset problem. In the imbalanced dataset problem, the prior probabilities of each class are extremely unequal or imbalanced. In addition, it is often extremely important to correctly classify the classes with the smallest priors. For example, in the simplest two class case, the class with the largest prior may represent non-cancerous patients, while the class with the smallest prior may represent cancerous patients. It is extremely important to classify cancerous patients accurately. Many classifiers when faced with imbalanced priors are overwhelmed by the imbalance, learning models which are not useful for discrimination.

Many other topics are covered by utility-based data mining, but they are outside the scope of this proposal.

One can view the work in this dissertation as a means of connecting topics within utility-based data mining that are typically studied separately. In particular,

work from the imbalanced dataset problem, cost-sensitive learning, and evaluation metrics are connected with different utilities (label acquisition costs and misclassification costs) in active learning. The resultant improvements to active learning are due to these connections.

1.3 Summary of Dissertation

The main goal of this dissertation is to improve the current capabilities of active learning. In particular, we will introduce a computationally efficient method of handling such things as points with different misclassification costs and points with different labeling costs. To do this, we will leverage techniques in utility-based data mining typically not connected to active learning.

In summary, the completed research presented in this dissertation accomplishes the following:

1. a new method of resampling for the imbalanced dataset problem
2. clarification of the relationship between resampling and cost-sensitive learning
3. a framework for analyzing evaluation metrics
4. a method for performing cost-sensitive active learning
5. a method for performing active learning for arbitrary evaluation metrics
6. active learning with non-uniform label acquisition costs, particularly spatially driven label acquisition costs

All of the above research allows for a method of improving active learning. In short, the completed research results in computationally efficient active learners that can handle points with different misclassification costs, that can handle different evaluation metrics, and that can handle different label acquisition costs. This

is important because one of the major limitations of supervised learning is that many domains require extremely large amounts of labeled data in order to build good classification models. While active learning can reduce the amount of labeled data required, one of the most computationally efficient active learning methods (i.e., uncertainty sampling) has no ability to handle different misclassification costs, different evaluation metrics, or different label acquisition costs. These aspects are very practically important in many domains, and this research fills this gap.

1.4 Outline

We discuss background and related work in active learning, cost-sensitive learning, imbalanced datasets, self-training, and evaluation metrics in chapter 2. We describe our own research in chapters 3 through 5. Finally, in chapter 6, we conclude the dissertation.

1.5 Notation

We will use the following notation throughout this dissertation. We will use lower-case letters for scalars and bold lower-case letters for vectors. Functions will be denoted as bold, italicized, lower-case letters. $\frac{d\boldsymbol{f}}{dx}$ denotes the derivative of a function $\boldsymbol{f}$ with respect to x while $\frac{\partial \boldsymbol{g}}{\partial x}$ denotes the partial derivative of a function $\boldsymbol{g}$ with respect to x. Matrices will be denoted as bold, capital letters while sets will be denoted as script, capital letters. In particular, we will use $\mathcal{U}$ to represent unlabeled training data and $\mathcal{L}$ to represent labeled training data in our work on active learning. In addition, let $|\mathcal{S}|$ be the number of points in a set $\mathcal{S}$.

For two-class problems, let constants n^+ and n^- represent the number of points in the positive and negative classes, respectively. Let n_{tp} represent the current number of true positives and n_{tn} represent the current number of true negatives.

Similarly, n_{fp} and n_{fn} will refer to the number of false positives and false negatives, respectively. For the multi-class case, the constant n^j represents the number of points from the jth class.

Chapter 2

Background and Related Work

In this section, we describe background and related work necessary for understanding this dissertation. We begin by discussing relevant problems from utility-based data mining and current solutions to those problems. We then discuss two problems that may at first seem unrelated: self-training and evaluation metrics used on supervised learning problems. These two topics will be tied together with utility-based data mining in chapter 4.

Finally, we describe the datasets used during experimentation throughout this dissertation. In particular, while the proposed solutions in this dissertation are not application specific, most of the active learning experiments were performed on hyperspectral data, so we will specifically address the task of classifying land cover types using hyperspectral data.

2.1 Relevant Problems and Current Solutions in Utility-based Data Mining

2.1.1 Active Learning

In active learning, the goal is to reduce the number of labeled training examples needed to train a classifier (see [Set09] for a recent survey). That is, given a classifier trained without active learning on a training set of size n_{train} points, the goal of active learning is to learn a classifier with comparable performance while using a training set of size significantly less than n_{train}. The active learner does this by selecting currently unlabeled points and having some expert provide labels for a subset of those points. In contrast, in the typical supervised learning scenario, a randomly chosen subset of data points is labeled. Since creating large amounts of labeled data is typically very expensive, active learning is extremely useful in reducing the amount of human time required to create a labeled dataset suitable for machine learning.

Active learning is typically studied in a "pool-based" setting, where there is a fixed pool of unlabeled points that the oracle can draw. Other settings (such as online settings where new points may be added to the pool of unlabeled points over time) are outside the scope of this dissertation.

Pool-based active learning is an iterative process where, on each iteration of active learning, the active learner chooses n points from the currently unlabeled pool of training data $\mathcal{U}$, presents these n points to an oracle for labeling, and then adds these n recently labeled points to the labeled training data $\mathcal{L}$. Roughly, the main difference between active learning approaches is how they pick points from $\mathcal{U}$ for labeling.

In this dissertation, we will be primarily concerned with techniques that have been used in empirical studies. In particular, we will contrast uncertainty

sampling methods [LC94] and loss reduction methods [RM01]. Recently, many theoretically driven approaches have been introduced (e.g., [BDL09] among others), but these are outside the scope of our dissertation. Instead, the goal is to improve the capabilities of uncertainty sampling, a widely used method in empirical studies which is computationally fast but is missing many properties of loss reduction approaches, another method widely used in empirical studies which is computationally slow.

In the loss reduction approach [RM01], the value of labeling a point in $\mathcal{U}$ is set equal to the expected reduction in loss for adding the point to $\mathcal{L}$. There are several benefits to using a loss reduction approach. In particular, the main advantage of using a loss reduction approach is that one can directly build a classifier useful for minimizing a certain loss or evaluation metric. For example, in [Mar05], it was shown how one can incorporate non-uniform label acquisition costs and misclassification costs into a loss reduction type setting.

However, the main drawback of loss reduction type approaches is computational efficiency. To obtain the expected reduction in loss for all points in $\mathcal{U}$, one needs to retrain the classifier $|\mathcal{U}| * n_c$ times, where n_c is the number of classes. This is particularly problematic for typical hyperspectral datasets, which have greater than 10 classes. While there are methods of performing loss reducing active learning more quickly such as subsampling $\mathcal{U}$ [RM01], computational efficiency is still a major issue. Without these speed-ups, $(|\mathcal{U}| * n_c) + 1$ classifiers need to be trained on each iteration of active learning, where, as mentioned, $|\mathcal{U}| * n_c$ classifiers are trained to find loss reduction scores and an additional classifier is trained on the union of previously labeled and newly labeled points.

In contrast, uncertainty sampling is a much more computationally efficient method of active learning which only requires one classifier to be retrained each iteration of active learning. That is, the only classifier that needs to be retrained is the classifier that trains on the union of previously labeled and newly labeled

points. In uncertainty sampling, some uncertainty score u_i is defined for each point $\mathbf{x_i}$. As the name implies, intuitively, the uncertainty score u_i corresponds to how "uncertain" the classifier is about the predicted class of some point $\mathbf{x_i}$. For example, if a classifier produces posterior probabilities, the uncertainty score is often defined to be inversely proportional to the variance of the posteriors or inversely proportional to the difference between the largest and second largest posterior probabilities (often referred to as the "margin").

Active learning results are typically plotted in the form of a "banana curve," a traditional format for presenting active learning results. Here, the horizontal axis corresponds to number of active learning iterations and the vertical axis corresponds to the error rate on the test set. We will use the banana curve (or variants of the banana curve) in chapters 4 and 5.

Finally, let us make a comment on active learning methods on spatial data, which we will address in chapter 5. Although many active learning strategies have been proposed during the last 15 years, there exist few algorithms that consider spatial characteristics of unlabeled samples. In [RGC08], the authors proposed a loss-reduction based active learning algorithm for hyperspectral data that adapts a classifier for spatial variation of spectral signatures. However, it does not take into account any form of varying label acquisition costs based on spatial data. An active learning algorithm to efficiently model spatial phenomena with Gaussian process has been proposed [KG07], but the algorithm is used to model spatially varying quantities and is not applicable to classification problems. We are unaware of any active learning studies that take spatially dependent label acquisition costs into account.

2.1.2 Cost-sensitive Learning

Cost-sensitive learning is a supervised classification problem where there is a non-uniform cost for misclassifying different data points. These costs may arise for a number of reasons. For example, in the realm of cancer detection, it is more important to detect patients with malignant strains of cancer than to accidentally run additional tests on a healthy patient. In credit card fraud detection, misclassifying a transaction as fraudulent incurs some small cost in the form of wasted time/annoyance on the part of the card holder, but allowing a fraudulent transaction could consist of a large cost in the form of stolen goods.

Researchers have looked at a number of ways of modeling costs in cost-sensitive learning. Perhaps the most common approach (described in [Elk01] is to define a cost matrix describing misclassification costs. In this formulation, a cost matrix $\mathbf{C}$ can be defined where $\mathbf{C}(i,j)$ is the misclassification cost for classifying a point with true class $Y = y_j$ as class $Y = y_i$. Typically, $\mathbf{C}(i,i)$ is set to zero for all i such that correct classifications are not penalized. In this case, the decision rule is modified (as discussed in [Elk01]) to predict the class y_i that minimizes $\sum_j P(Y = y_j|\mathbf{X} = \mathbf{x})C(i,j)$, where $\mathbf{x}$ is the data point currently being classified.

The goal in cost-sensitive learning is to minimize the total cost $\sum_{i,j} \mathbf{N}(i,j)\mathbf{C}(i,j)$ where $\mathbf{N}(i,j)$ is the number of data points from class j classified as class i. That is, the goal is to minimize the total or expected cost of misclassification incurred on some test set. Note that a classifier that minimizes the total cost for a cost matrix $\mathbf{C}$ also minimizes the total cost for cost matrix $\mathbf{C}' = a\mathbf{C} + b$, where a and b are scalars. In addition, $\mathbf{C}(i,i)$ is typically set to zero for all i such that correct classifications are not penalized.

Many solutions involve turning specific classifiers into cost-sensitive classifiers. For example, a Bayesian classifier can be easily modified to predict the class y_i that minimizes $\sum_j P(Y = y_j|\mathbf{X} = \mathbf{x})C(i,j)$ as we will show in the next chap-

ter. This modification involves shifting the decision boundary by some threshold. SVMs can be modified as described in [MBJ99], where instead of a single parameter controlling the number of empirical errors versus the size of the margin, a separate parameter for false positives and a second parameter for false negatives are used.

Other, classifier agnostic approaches have been proposed such as Metacost [Dom99], cost-proportionate rejection sampling [ZLA03] or costing [ZLA03]. Most relevant to our work is cost-proportionate rejection sampling. Cost-proportionate rejection sampling is based on a theorem that describes how to turn any classifier which reduces the number of misclassification errors into a cost-sensitive classifier and involves using rejection sampling to create a new training set from the original training set. In this case, however, there is a single misclassification cost for each point (versus a cost matrix C for all points). Given that the kth data point in the training set has some possibly unique misclassification cost c_k, the kth data point in the training set has probability $\frac{c_k}{z}$ of being included in the resampled training set, where z is a user-defined parameter (e.g., let z equal the largest value of c_k in the training set in order to maximize the expected size of the resampled dataset).

2.1.3 Imbalanced Datasets

In machine learning, a supervised classifier learns a model from labeled training data that allows it to discriminate between data points from different classes. In many studies in the past, the prior probabilities of each class within the training data were approximately equal. For example, in classic benchmark datasets such as Iris or 20-Newsgroups, each class has essentially equal class priors.

However, in many real life applications, datasets have class priors which are far from equal. A classic example of this is cancer detection. In cancer detection, the majority of cells are non-cancerous, while a small percentage of cells are actually cancerous. The goal is to detect these cancerous cells as accurately as possible.

However, because of the imbalanced priors, many standard classifiers have trouble in this domain. This is an example of the imbalanced dataset problem, a classification task where the prior probabilities of all classes are highly unequal or imbalanced. While easy to learn a classifier with high accuracy on the non-cancerous cells (e.g., by always predicting non-cancerous), the goal is to train a classifier which is able to accurately detect and discriminate between cancerous cells and non-cancerous cells. Other example problems that exhibit class imbalance class include fraud detection [CS98] [PAL04], keyword extraction [Tur00], oil-spill detection [KHM98], direct marketing [LL98], and information retrieval [LC94].

Typically, the imbalanced dataset problem is posed as a two-class problem where the minority class has an extremely small prior when compared to the majority class. We claim that the imbalanced dataset problem is related to cost-sensitive learning since it is usually more important to classify the minority class. That is, there is a greater cost for misclassifying the minority class when compared to the cost for misclassifying the majority class. This cost is not typically known, nor is it necessary that these costs be well-defined. All that needs to be known is that the cost of misclassifying the minority class is greater. If this were not the case, the trivial classifier which always predicted the majority class might indeed be the best possible classifier on this data. However, this is not acceptable in domains such as cancer detection where there is some unknown but higher cost for misclassifying the minority class. Thus, the imbalanced dataset problem can be seen as a specific type of cost-sensitive learning problem where: 1) one or more classes have extremely small prior probabilities with respect to the prior probabilities for other classes and 2) the classes with small priors also have much higher misclassification costs with respect to the misclassification cost for classes with larger priors.

Solutions to solving the imbalanced dataset problem include cost-sensitive learners and resampling. Since the imbalanced dataset problem involves a minor-

ity class with a higher misclassification cost than the majority class, cost-sensitive learning can be readily applied by defining some cost for the minority class that is greater than some cost for the majority class. Thus, any cost-sensitive learning algorithm can be applied to the imbalanced dataset problem.

Resampling is a popular technique to handle imbalanced datasets because it is both simple and effective. The idea behind resampling is to increase the prior probability of the minority class during training by either removing points in the training set belonging to the majority class (undersampling), by adding minority class points to the training set (oversampling), or by doing both. The two most basic techniques are random undersampling and random oversampling. In random undersampling, points are randomly removed from the majority class until the desired level of balance between the minority and majority class priors has been reached. In random oversampling, points are randomly selected from the minority class and duplicated. Each approach has several known drawbacks. For example, random undersampling removes potential information from the training set. In contrast, random oversampling merely duplicates existing points, possibly leading to overfitting. Moreover, random oversampling may drastically increase the memory requirements of a dataset, making it unwieldy in practice.

Despite these drawbacks, random undersampling and random oversampling work well in empirical studies. In addition, while many resampling methods have been proposed in addition to random oversampling and random undersampling, surprisingly, few have consistently outperformed both of these simplistic approaches. Two techniques that have shown promise over simple random oversampling and undersampling are SMOTE and cost-proportionate rejection sampling.

SMOTE (Synthetic Minority Oversampling TEchnique) is an oversampling technique that adds synthetic examples of minority class points to the training set [CBHK02]. That is, instead of duplicating existing minority class data points, it

adds datapoints artificially created from known minority class points. In order to do this, SMOTE first finds the distance between each minority class point and its k nearest neighbors in the minority class. Given a minority class point, one of its k nearest neighbors is randomly selected, and a point on the line segment joining the two points is randomly selected as the new, synthetic point.

Cost-proportionate rejection sampling, which was described above, can be seen as a cross between resampling and cost sensitive classifiers. Interestingly, under certain conditions, random undersampling and cost-proportionate rejection sampling are equivalent methods. Instead of specifying a separate cost c per data point, many cost-sensitive approaches assume a constant cost for misclassifying one class as another. As discussed above, for a k-class problem, this means a k by k cost-matrix $\mathbf{C}$ can be defined. In the standard framework for studying imbalanced data, k is equal to 2 (the minority class and the majority class), and there is some constant cost for misclassifying the minority class ($\mathbf{C}(1,2)$) and a constant cost for misclassifying the majority class ($\mathbf{C}(2,1)$). While the exact costs may be unknown, it is known that $\mathbf{C}(1,2) > \mathbf{C}(2,1)$. In this case, cost-proportionate rejection sampling boils down to random undersampling with some probabilistically determined rate of resampling. If we seek to maximize the size of the resampled dataset by setting z equal to $\mathbf{C}(1,2)$, then the expected number of majority class points will be $\frac{\mathbf{C}(2,1)}{\mathbf{C}(1,2)}$ times the size of the original majority class. Thus, under common assumptions when empirically classifying imbalanced datasets, cost-proportionate rejection sampling is equivalent to random undersampling with a probabilistically determined number of points to remove from the majority class.

In [Elk01], a mathematical description of cost-sensitive learning versus resampling is given. Empirical comparisons have also been made between cost-sensitive approaches and resampling. Most notably, [MZW05] found that there was very little difference in the performance between cost-sensitive learners and resampling. This

is consistent with older studies which show that similar effects can be accomplished by changing probability thresholds on the outputs of probabilistic classifiers, varying resampling rates, and varying costs (e.g., [Mal03]). In chapter 3, we will first discuss a new approach to resampling and then re-examine the connection between resampling and cost-sensitive learning.

2.2 Self-training

As opposed to supervised learning which builds a model only using labeled data points, semi-supervised learning is a method of using both labeled and unlabeled data to build models for tasks such as classification. There are a wide variety of semi-supervised approaches, and a description of these is far beyond the scope of this dissertation (see [Zhu05] for a good survey).

Self-training is a popular semi-supervised learning algorithm. In self-training, the classifier produces (predicted) class labels for unlabeled data, and uses the unlabeled data with predicted labels as augmented training data to re-train the classifier itself. The Expectation-Maximization (EM) algorithm can be thought of as a self-training algorithm [Zhu05].

Semi-supervised learning and active learning can both be useful approaches in supervised learning problems with minimal amounts of labeled data. Roughly, semi-supervised learning uses the current set of labeled data augmented with a large amount of unlabeled data while active learning attempts to increase the size of the labeled data set. We will show in chapter 4 that self-training can be combined with uncertainty sampling to create an effective method for building cost-sensitive classifiers via active learning.

2.3 Evaluation Metrics

In this section, we describe work on analyzing common evaluation metrics used to determine performance on supervised learning problems. While this work may seem unrelated to utility-based data mining, particularly the problems described earlier, we will show in chapter 4 that a large class of evaluation metrics are indeed related to cost-sensitive learning.

In [Fla03], the authors argue that one way to characterize an evaluation metric is by examining its isometric curves in ROC space. In 2-D ROC space, the horizontal axis corresponds to false positive rate (i.e., $\frac{n_{fp}}{n^-}$) while the vertical axis corresponds to true positive rate (i.e., $\frac{n_{tp}}{n^+}$). The isometric curves of a metric are defined by setting the metric equal to some constant. The shape of these curves in ROC space can be used to characterize the evaluation metric. Moreover, a value called the effective skew ratio can be used to summarize these curves. The effective skew ratio is the slope of an isometric curve.

[VO00], [CNM04], and [Fla03] describe potential methods of comparing evaluation metrics. In [VO00], the difference between two metrics is defined with respect to some value of n_{tp} and n_{tn}. However, it is not clear how to choose these values of n_{tp} and n_{tn} since choosing different values of n_{tp} and n_{tn} (e.g., close to n^+ or close to 0) produce very different values in the distance between two metrics. [CNM04] also compares metrics empirically. In [CNM04], the authors run an extremely large volume of experiments using different datasets, classifiers, and parameter settings and use several common metrics to evaluate their results. Using the metrics as datapoints and using the results for each of the experiments as a feature, the authors project the metrics into a lower dimensional manifold using MDS. While the results of this approach are interesting, it does not explain why certain metrics are close to each other in the lower dimensional manifold; moreover, the approach does not suggest a simple method of characterizing future metrics besides running additional

experiments. A more sophisticated, analytical method that is not coupled with the results of a particular set of experiments is described in [Fla03], which shows that metrics which are equivalent have equivalent effective skew ratios. Unfortunately, the converse is not true; metrics with the same effective skew ratio are not necessarily equal.

Besides characterizing a metric's skew and being able to directly compare metrics, many more questions about metrics remain as open research questions. We will show one way in which work such as that in [Fla03] can be used in conjunction with cost-sensitive learning to build a computationally efficient active learning approach that is able to optimize specific evaluation metrics.

2.4 Datasets

During experimentation, we use many datasets from many different domains throughout this dissertation. Since many of these datasets (particularly hyperspectral data) are used in multiple sets of experiments, we will describe these datasets here.

2.4.1 Hyperspectral Datasets

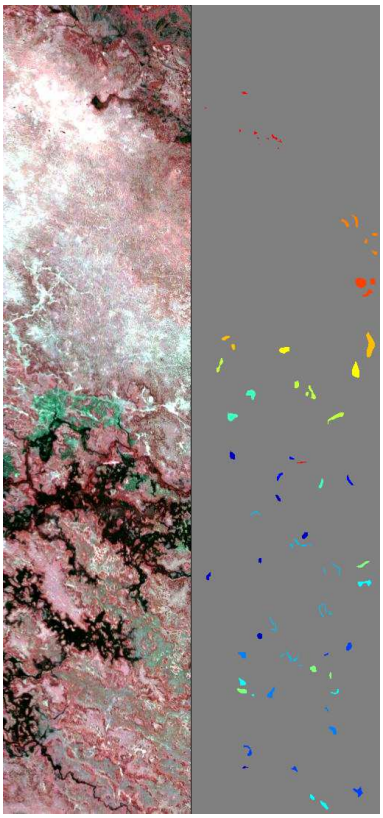
Land cover classification by hyperspectral image (HSI) data analysis has become an important part of remote sensing research in recent years [Lan02][JL98][MB04]. Compared to conventional multi-spectral images where each pixel usually contains tens of bands, pixels in hyperspectral images usually consist of more than a hundred spectral bands, providing fine-resolution spectral information. Classification techniques used for this application should be able to handle high-dimensional high-resolution data and a fairly high number of classes. Obtaining ground truth is another challenge, since HSI can cover very large areas but it is not usually possible to obtain highly accurate class labels for all locations in the image.

We use hyperspectral images taken from two geographically different loca-

Table 2.1: The class names in the KSC and Botswana datasets

KSC Dataset	
Class number	Class Name
1	scrub
2	willow swamp
3	cabbage palm hammock
4	cabbage palm/oak hammock
5	slash pine
6	oak/broadleaf hammock
7	hardwood swamp
8	graminoid marsh
9	spartina marsh
10	cattail marsh
11	salt marsh
12	mud flats
13	water
Botswana Dataset	
Class number	Class Name
1	water
2	hippo grass
3	floodplain grasses 1
4	floodplain grasses 2
5	reeds1
6	riparian
7	firescar2
8	island interior
9	acacia woodlands
10	acacia shrublands
11	acacia grasslands
12	short mopane
13	mixed mopane
14	exposed soils

tions: NASA's John F. Kennedy Space Center(KSC) [Mor02] and the Okavango Delta in Botswana [HCCG05]. We will call the two datasets the KSC and Botswana datasets, respectively. The goal of these datasets is to correctly classify types of land. Table 2.1 contains a list of the classes in the KSC and Botswana datasets, while Figure 2.1 shows images of Botswana and KSC with their corresponding class maps. The grey areas in the figure denote areas which have "no label." As can be



(a) Botswana

(b) KSC

Figure 2.1: Botswana and KSC images with class maps

seen in this figure, only a small fraction of the entire region actually has class labels.

The KSC dataset was acquired by NASA Airborne Visible/Infrared Imag-

ing Spectrometer (AVIRIS) and originally consisted of 242 bands. After removing noisy bands, only the remaining 176 bands are used for classification. There are 13 different land cover types including water and mixed classes, which causes some difficulties in classification. The hyperspectral image used for experiments has 512×614 pixels with 18m spatial resolution.

The Botswana dataset was obtained from the Okavango Delta by the NASA EO-1 satellite with the Hyperion sensor on May 31, 2001. The acquired data originally consisted of 242 bands, but only 145 bands are used after preprocessing. The area used for experiments has 1476×256 pixels with 30m spatial resolution, with 14 different land cover classes.

In all experiments, we preprocess the data in two ways that are known to be effective for classifying hyperspectral data. First we utilized spatial information in addition to the spectral information, via the max-cut algorithm described in [CCG07], where a pixel's feature vector is augmented with features from neighboring pixels whose spectral features are similar to the pixel of interest. The max-cut algorithm takes advantage of unsupervised information and provides a way to identify pixels that are close both in physical and spectral spaces, and produces more accurate and stable classification results for spatial data. Second, we applied best-basis feature reduction which exploits the high correlation between certain adjacent spectral bands, and is tailored for hyperspectral data analysis [KGC01].

Finally, we should note that the KSC and Botswana datasets can be modeled well with Gaussians, as shown empirically in [RGC08].

2.4.2 Text Datasets

We use a variety of text datasets from various sources. These datasets were drawn from the twenty newsgroups dataset [AN07] and sample data included in the CLUTO toolkit [Kar02]. As the name suggests, the twenty newsgroups dataset consists of

data taken from twenty different newsgroups on topics such as religion, computers, sports, and science. The data included with the CLUTO toolkit comes from several past studies in information retrieval including TREC (http://trec.nist.gov), OHSUMED [HBLH94], and WebAce [HBG+98]. The datasets from TREC are all newspaper stories from either the LA Times (la12 dataset) or San Jose Mercury (hitech, reviews, sports datasets) classified into different topics. The ohscal dataset contains text related to medicine, while the k1b dataset contains documents from the Yahoo! subject hierarchy. Instead of the original data included with the CLUTO toolkit, we use the datasets available at http://www.ideal.ece.utexas.edu/data/docdata.tar.gz, which have some additional preprocessing as described in [ZG03].

2.4.3 Other Datasets

Finally, we also use various low-dimensional datasets from the UCI dataset repository from a number of domains. The main characteristic shared by these datasets is that they consist of a relatively low-number of numerical features ("low" as compared to hyperspectral or text data).

These datasets include: pima indian, wine, breast cancer wisconsin diagnostic (wdbc), breast cancer wisconsin prognostic (wpbc)[1], page-blocks (using "non-text" versus "rest" as our binary class problem), and ionosphere. The features of all of the dimensions for wine and breast cancer wisconsin diagnostic (wdbc) pass the Kolmogorov-Smirnov test for normality for a p-value of 0.05; most of the features of the breast cancer wisconsin prognostic (wpbc) dataset also pass the Kolmogorov-Smirnov test for normality. In contrast, the majority of the features of the remaining datasets did not. We use both features that passed and did not pass the Kolmogorov-Smirnov test in our experiments, so the assumption that Gaussians can be used to model the various datasets does not hold very well on some of the datasets.

[1]Note that the two breast cancer datasets are separate datasets in the UCI dataset repository, and not the same dataset used for two different tasks in our experiments.

Chapter 3

Imbalanced Datasets: Resampling and Cost-sensitive Learning

3.1 Problem Description

In active learning, there is a limited amount of labeled data on which to train. The active learner picks points to label and provides them to an oracle in order to be labeled. However, this oracle (typically a human) incurs non-trivial time, effort, and other costs in order to label points. One idea, then, is to speed up active learning by obtaining labeled points from some other oracle that is faster.

For inspiration, we look to the realm of imbalanced datasets. In the imbalanced dataset problem, there is both a relative and absolute scarcity of labeled data, particularly from one class. The relative scarcity is simple to understand: the prior of one class is, by definition, much smaller than the prior of the other class in the imbalanced dataset problem. However, it is also typically the case that there is also absolute scarcity in the minority class. That is, there are very few points in the

minority class.

Thus, one idea is that methods used in the imbalanced dataset community to address both relative and absolute data scarcity can be leveraged to handle active learning, a problem where there is also absolute data scarcity. In this section, we first study the commonly used technique of resampling to handle imbalanced datasets. In particular, we introduce a new method of resampling which is superior to other methods in the realm of text classification. We then study the relationship of resampling and cost-sensitive learning. This research clarifies when resampling and cost-sensitive learning are equivalent. We close this section with a discussion of how this research can be leveraged for active learning.

3.2 Generative Oversampling

In problems with high class imbalance, without accounting for imbalanced priors, a classifier may learn to always predict the majority class. Given that the cost of misclassifying minority class points may be extremely high, a classifier that always predicts the majority class is not acceptable. Many approaches have been proposed and studied in order to handle imbalanced problems (see [VR05] for a recent survey). As discussed in Chapter 2, two solutions to the imbalanced dataset problem are cost-sensitive learning and resampling the training set.

In this section, we focus on resampling for a variety of reasons. First, resampling is a very common technique used to handle class imbalance. Empirically, results using resampling have been shown to be competitive [MZW05] or even nearly identical (for certain classifiers) [Mal03] to results using cost-sensitive learning. Resampling methods are also extremely simple to implement in practice. Resampling is also a wrapper approach that is classifier agnostic. Finally, we will examine the connection between resampling and cost-sensitive learning in the next section.

Previously introduced resampling techniques were discussed in Chapter 2.

In this section, we introduce a resampling technique called generative oversampling. In generative oversampling, artificial data points are generated from an assumed probability distribution whose parameters are learned from the training data. Generative oversampling takes advantage of the fact that in many domains, appropriate families of probability distributions are known that can model data well. Generative oversampling is simple and straightforward to implement. Despite this simplicity, we empirically show that generative oversampling can outperform popular resampling techniques.

In this section, we first discuss why the use of a probability distribution with resampling is natural and well motivated. We then describe the generative oversampling algorithm which creates completely new, artificial data points via a chosen probability distribution. Finally, we make some comments and observations about generative oversampling.

3.2.1 Motivation

For the sake of discussion, consider the following two-class classification problem where the data set $\mathcal{X}$ (as well as specific training set $\mathcal{X}_{train}$ and test set $\mathcal{X}_{test}$) is the union of points from sets $\mathcal{P}$ and $\mathcal{Q}$ where:

1. $\mathcal{P}$ is a set of points from one class; these points are drawn from some unknown distribution

2. $\mathcal{Q}$ is a set of points from the second class; these points are drawn from a second unknown distribution

3. and $\lambda = \frac{|\mathcal{P}|}{|\mathcal{X}|}$; that is, λ is the prior probability that a point is from set $\mathcal{P}$.

The distributions that generate $\mathcal{P}$ and $\mathcal{Q}$ can be arbitrarily complicated, and the goal of a two-class classifier is to learn to distinguish between points from $\mathcal{P}$ and points from $\mathcal{Q}$. In many domains, the family of probability distributions suitable for

modeling $\mathcal{P}$ and $\mathcal{Q}$ is known. For example, many low-dimensional datasets can be modeled by mixtures of Gaussian distributions (e.g., [MM04]), whereas text datasets can be modeled as mixtures of multinomial distributions (e.g., [MN98]).

A two-class, imbalanced dataset problem can be phrased in terms of $\mathcal{P}$ and $\mathcal{Q}$ by letting $\mathcal{P}$ represent points from the minority class and $\mathcal{Q}$ represent points from the majority class. $\mathcal{X}$ becomes imbalanced when the minority class prior λ is much less than $1 - \lambda$. The goal of resampling is to either increase the number of points drawn from the distribution that produces $\mathcal{P}$ or decrease the number of points drawn from the distribution that gives rise to $\mathcal{Q}$. That is, ideally, oversampling techniques [1] would add points to training set $\mathcal{X}_{train}$ that have been drawn directly from the distribution that originally produced $\mathcal{P}$.

Unfortunately, we typically cannot draw additional points from this original distribution since it is unknown. Instead, some facsimile for drawing from this unknown distribution is used instead (such as random oversampling or SMOTE). However, while we cannot obtain new points from the original distribution, we can attempt to model the distribution that produced $\mathcal{P}$ and create new points from this model. We describe such an algorithm in the next section.

3.2.2 The Generative Oversampling Algorithm

Generative oversampling can be used in any domain where there exist probability distributions that model the actual data distributions well. Generative oversampling works as follows:

1. a probability distribution is chosen to model the minority class
2. based on the training data, parameters for the probability distribution are learned

[1] For a discussion of undersampling and its effect on the training set in terms of $\mathcal{P}$ and $\mathcal{Q}$, see [ZLA03].

3. artificial data points are added to the resampled data set by generating points from the learned probability distribution until the desired number of minority class points in the training set has been reached.

The idea behind generative oversampling is simple, straightforward, and is a natural approach in data mining and machine learning. Indeed, the idea of creating artificial data points through a probability distribution with learned parameters has been used for many other applications (e.g., [MM04] for creating diverse classifier ensembles, [BCNM06] for model compression). Surprisingly, however, it has not been used with respect to resampling techniques.

3.2.3 Generative Oversampling for Text Datasets

The multinomial distribution has been successfully used with the bag of words representation in text mining (e.g., [MN98]). We now present a specific example of generative oversampling for text datasets that assumes a multinomial distribution. While other distributions have been applied to model text, we choose the multinomial distribution in our experiments because it has been well studied and because multinomial naive Bayes has been widely applied with success. A multinomial model with Laplace smoothing is learned from the minority class training data such that the probability of a word appearing in a document from the minority class is given by:

$$p(w_i|x_j \in \mathcal{P}_{train}) = \frac{l + \sum_{x_j \in \mathcal{P}_{train}} x_j(i)}{\sum_{a=1}^{n_v}(l + \sum_{x_b \in \mathcal{P}_{train}} x_b(a))} \tag{3.1}$$

where $\mathcal{P}_{train}$ represents the set of minority class points in the training set, w_i represents the ith word in the vocabulary, n_v represents the number of words in the vocabulary, $\mathbf{x_j}$ represents the jth document in $\mathcal{P}_{train}$, $x_j(i)$ is the number of times w_i occurs in document $\mathbf{x_j}$, and l controls the amount of smoothing. Note that, if $l > 0$, artificial minority class documents created using generative oversampling have a non-zero probability of containing words not contained in the minority class

during training.

An artificial document is created by drawing from the learned multinomial distribution n_{ave} times, where n_{ave} is the average length of the minority class documents in the training set.

3.2.4 Discussion

One possible drawback to generative oversampling is that there may not be enough data to properly learn the parameters of the chosen probability distribution. This is related to the difference between relative scarcity and absolute scarcity, two problems that are often conflated when dealing with class imbalance. Absolute scarcity describes a problem where there is simply a small number of data points from a particular class, whereas relative scarcity describes the problem where the prior probability of a class is small relative to the priors of the other classes in the dataset. Intuitively, generative oversampling appears better at handling classes that are relatively scarce but not absolutely scarce. However, absolute scarcity is a problem for not only generative oversampling, but other techniques as well; that is, handling data that is absolutely scarce is an important and open research question that is not addressed well by current approaches.

Generative oversampling for text using a multinomial distribution also has the following two amenable properties as long as some smoothing is used (i.e., $l > 0$). Firstly, generative oversampling has the capability of adding new minority points outside of the convex hull inscribing the original minority class points in the training set. This is desirable in many cases; for example, a well known problem with SVMs and imbalanced data is that the separating hyperplane is often pushed towards the minority class. By expanding the size of the minority class convex hull, the separating hyperplane is shifted away from the minority class. Secondly, when using generative oversampling, new points can include words not seen in the minority

class points in the training set, a feature which discourages overfitting the minority class. The capabilities to expand the minority class convex hull and to allow new, resampled points to take on new words not seen in the training set are due to the Laplace smoothing parameter l. In contrast, neither random oversampling, random undersampling, or SMOTE are able to add points outside of the convex hull or allow new words not seen in the minority class points in the training set.

3.2.5 Experiments

In this section, we empirically test generative oversampling against three other resampling methods: random oversampling, random undersampling, and SMOTE. We use SVMs in the domain of text classification in our empirical tests. We show that both resampling (particularly generative oversampling and random undersampling) and adjusting the tradeoff between the size of the margin and the number of support vectors can improve the performance of SVMs when classifying imbalanced data. Moreover, results obtained by tuning the tradeoff between the margin and support vectors can be improved even further through generative oversampling. We empirically show that generative oversampling works best when used with Laplace smoothing. Finally, we show that generative oversampling is robust to choosing how many resampled data points to add to the training set whereas the performance of other resampling methods (particularly random undersampling) can be highly dependent on the degree of resampling.

Below, we first describe the datasets used in our experiments, then describe our general experimental setup, and finally discuss our results.

Datasets

We test our resampling methods using text mining as an example domain. We use several of the text datasets described in Chapter 2, including the hitech, k1b, la12,

Table 3.1: Dataset characteristics

Dataset	Num min class pts	Min class prior(λ)
hitech	116	0.0504
k1b	60	0.0256
la12	521	0.0830
ohscal	709	0.0635
reviews	137	0.0337
sports	122	0.0142

ohscal, reviews, and sports datasets. All text datasets were converted into a standard bag of words model. In addition, we used TFIDF weighting and normalized each document vector with the L2 norm after resampling. Finally, since the imbalanced dataset problem is typically posed as a two-class problem, we converted each of our text datasets (which have multiple classes) into two-class problems. For each dataset, we chose the smallest class in each dataset as the minority class, and aggregated all other classes to form the majority class.

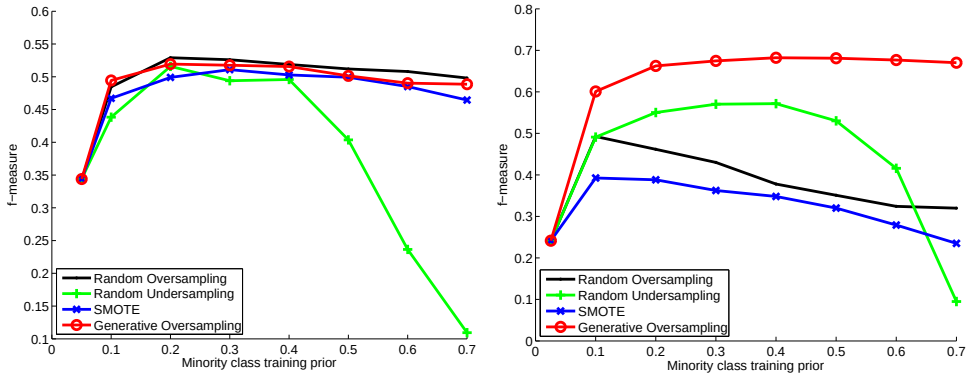
Details about these datasets are given in Table I, including the number of minority class points in the data and the value of the minority class prior λ.

Experimental Setup

For each dataset in our empirical evaluation, we apply the following basic steps:

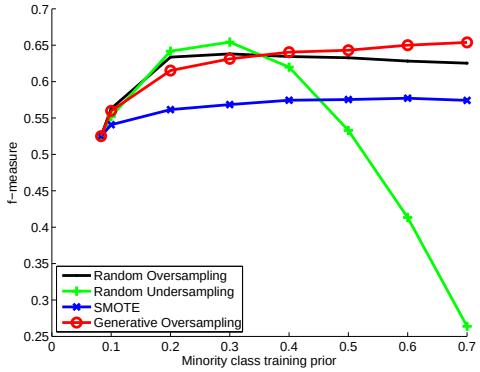
1. Divide the data into training and test sets
2. Resample training data
3. Apply SVM classifier and evaluate

For each dataset, we create ten different training and test splits by randomly selecting 50% of the data using stratified sampling as the training set and the rest as the test set. We test generative resampling with smoothing parameter $l = 1$ against three other resampling methods: random oversampling, random undersampling, and

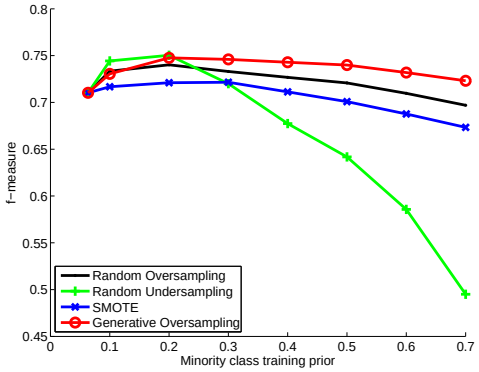


(a) hitech dataset

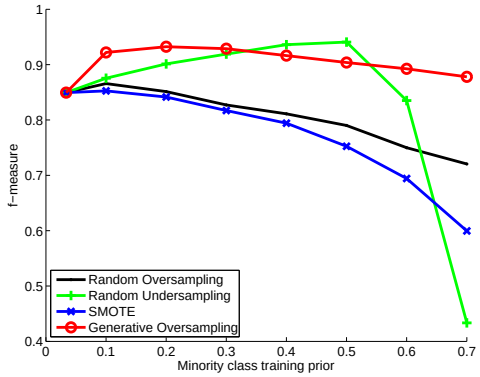
(b) k1b dataset



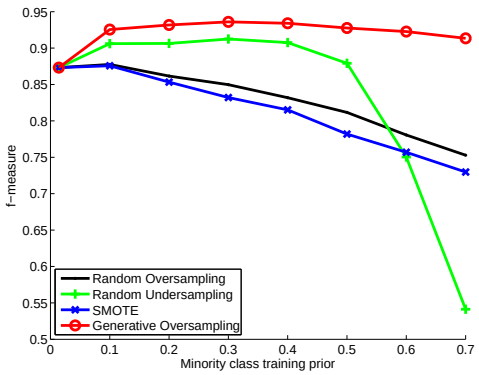
(c) la12 dataset



(d) ohscal dataset



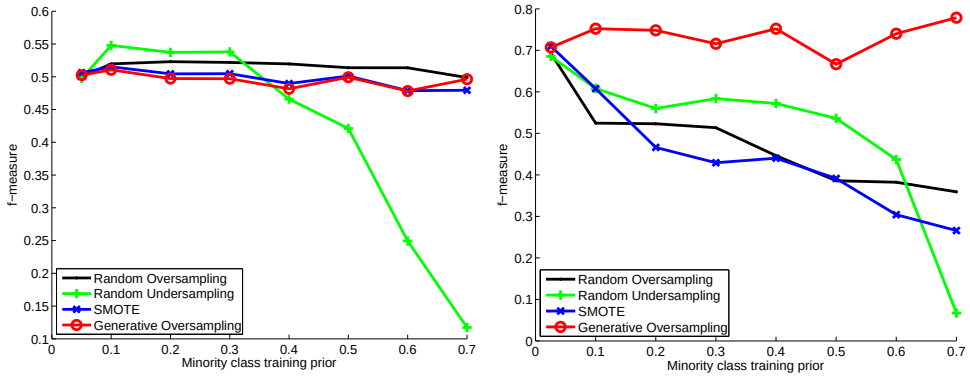
(e) reviews dataset



(f) sports dataset

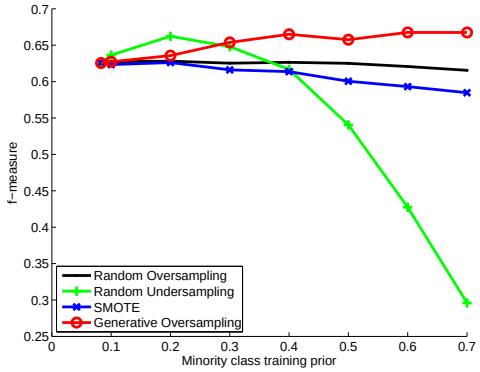
Figure 3.1: f-measure vs minority class training prior

SMOTE [CBHK02]. Note that since SMOTE requires a distance metric, we use 1-cosine similarity as the distance metric since 1-cosine similarity has shown to be a good metric for text data [DFG01]. SMOTE also requires a parameter to control the

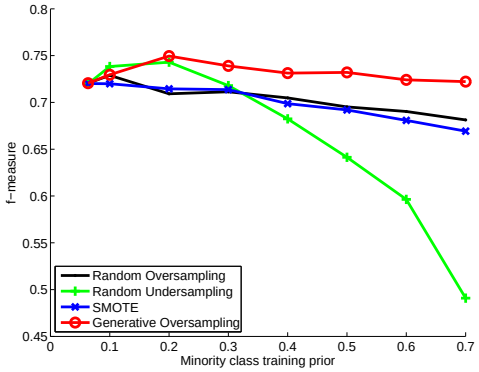


(a) hitech dataset

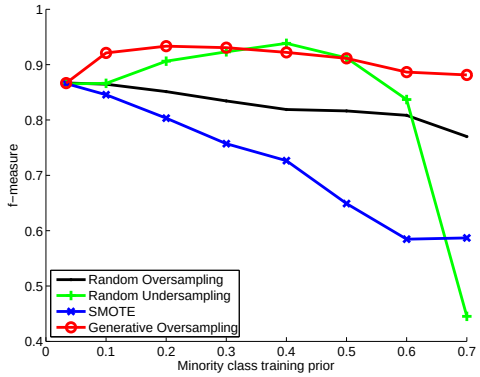
(b) k1b dataset



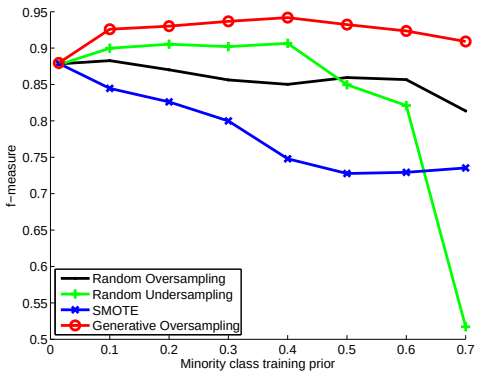
(c) la12 dataset



(d) ohscal dataset



(e) reviews dataset

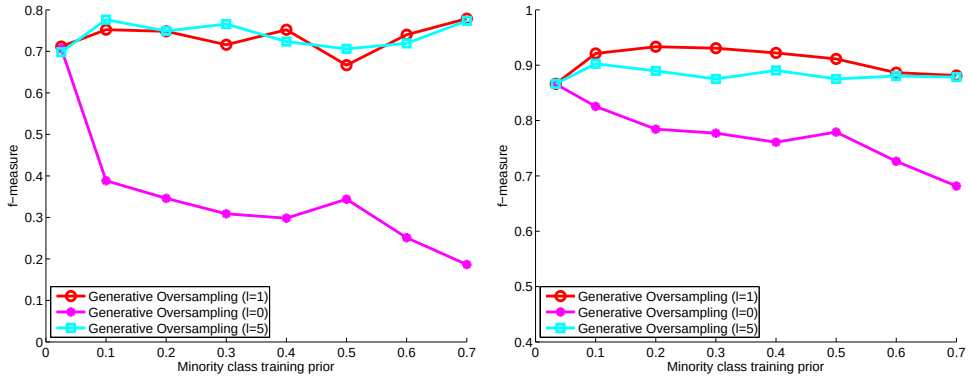


(f) sports dataset

Figure 3.2: f-measure vs minority class training prior while tuning C

number of nearest neighbors to consider when resampling; we use 5 as in [CBHK02].

In addition, we also perform experiments where we compare generative oversampling with different levels of smoothing. In particular, we try parameter $l = 1$,



(a) k1b dataset (b) reviews dataset

Figure 3.3: Example results with different levels of smoothing

$l = 0$ (no smoothing), and $l = 5$.

When resampling, one controls the percentage of the training set comprised of minority class points. We will call this the resampled minority class training prior. A resampled minority class training prior of $p\%$ means that after resampling, $p\%$ of the training set is comprised of minority class points. In our experiments, we vary the resampled minority class training prior between the natural prior λ and 70%. Note that when the resampled minority class training prior is equal to λ, no actual resampling has been performed.

We use an SVM with a linear kernel, which has been shown to work well in text classification [Yan99]. Specifically, we use the SVM-light implementation by Joachims [Joa98]. Note that, for SVMs, there is an additional parameter used to control the trade-off between the margin and training errors. In SVM-light this is the parameter C [2]. Adjusting C can affect the location of the separating hyperplane. In SVM-light, one can either specify C or use a default value of C estimated from the data. We run experiments using both settings. When specifying C, we perform a search for the best value of C by using a validation set; we further split each training

[2]Not to be confused with a cost matrix $\mathbf{C}$; note we use a capital letter to represent a scalar here, which is an abuse of our notation; we do this because C is typically represented as a capital letter by convention

set into a smaller training set (70% of initial training set) and validation set (the remaining 30% of the training set) and search for the best value of C between 2^{-6} and 2^{6}. Note that this tuning is done separately for every experiment (i.e., once for every combination of dataset, training set split, resampled minority class prior, and resampling method).

We use f-measure as our evaluation metric. Besides being a popular metric in information retrieval, f-measure has been used in other past studies on imbalanced data (e.g., [ZWS04]) and, unlike metrics like accuracy, is considered robust against class imbalance. ROC is another popular metric used to study imbalanced data, but we choose f-measure because of its prevalent and standard use in evaluating text classification.

In summary, we perform three sets of experiments. In the first set, we compare the effect of generative oversampling (with $l = 1$), random oversampling, SMOTE, and random undersampling on SVMs without tuning the trade-off between the size of the margin and the number of training errors. In the second set of experiments, we perform the same experiments while tuning the trade-off between the size of the margin and number of training errors. Finally, in the third set of experiments, we show that generative oversampling works well when used with Laplace smoothing by comparing generative oversampling with $l = 0$, $l = 1$, and $l = 5$.

3.2.6 Results

Figure 3.1 plots our results comparing generative oversampling, random oversampling, SMOTE, and random undersampling on each of the six datasets when all default parameters for linear SVMs are used. Figure 3.2 shows the results when the tradeoff between margin size and number of training errors is tuned via a validation set. The plots show f-measure as a function of resampled minority class training prior.

Effects on Location of the Separating Hyperplane

Generative oversampling potentially increases the size of the convex hull surrounding the minority class by producing artificial data points that occur both inside and outside of the original convex hull inscribing the minority class points in the training set. Random undersampling potentially reduces the size of the convex hull surrounding majority class training points by removing points from the majority class. Thus, both generative oversampling and random undersampling can be effective at moving the decision boundary found by a linear SVM, whereas random oversampling and SMOTE merely add points in the existing minority class convex hull.

We see the empirical effects of this in both figures 3.1 and 3.2 [3]. In figure 3.1, where the parameter C has not been tuned, we see that in all but the hitech datasets, either generative oversampling or random undersampling produce the highest f-measure. In particular, generative oversampling is either the clear winner (k1b, sports) or is extremely competitive with random undersampling.

Tuning the parameter C can also affect the location of the separating hyperplane. Thus, we see in figure 3.2 that the results with the natural prior are much higher than the results with the natural prior without tuning in figure 3.1. In particular, note that the results using the natural prior (even after tuning C) can be improved upon by using generative resampling or random undersampling. In comparison, there is minimal or no improvement using random oversampling or SMOTE as compared to using the natural prior if C is tuned. In fact, using random oversampling or SMOTE often does worse than simply using the natural prior. Thus, regardless of whether C is tuned, generative oversampling and random undersampling can both be used to improve upon results using the natural prior.

[3] Note that the vertical axes of all graphs are scaled differently in order to maximize visualization within each graph; however, to facilitate comparisons, the vertical axis of graphs corresponding to the same dataset are scaled identically.

Finally, Figure 3.3 contains sample results where we compare generative oversampling with different levels of smoothing (note that all of these experiments were run while tuning for C). The results in figure 3.3 show two general cases which are representative of performance on all the datasets: either results with $l = 5$ and $l = 1$ are comparable, or results with $l = 1$ are better than results with $l = 5$. With $l = 0$, generative oversampling does very poorly (often worse than random oversampling or SMOTE). When $l = 0$, generative oversampling no longer creates points outside of the original convex hull, supporting our argument that the reason that generative oversampling (with $l = 1$) performs well is due to its ability to expand the original convex hull of the minority class. For $l = 5$, the results obtained with generative oversampling are clearly worse than generative oversampling with $l = 1$ on two of the datasets. On the other datasets, generative oversampling with $l = 1$ and $l = 5$ are comparable, although $l = 5$ is slightly better at low minority class training priors. The exact reasons for the difference in performance between $l = 1$ and $l = 5$ is an open question.

Robustness to Choice of Minority Class Training Prior After Resampling

Unfortunately, the exact minority class training prior to use may be unknown. As many authors have noted in the past, using a completely balanced training set (i.e., minority class training prior of 0.5) rarely (never in our experiments) yields the best results. If there is an exact, known misclassification cost, then previous studies ([Elk01], [ZLA03]) offer a guidance on the minority class training prior to use for the best results. In fact, we will examine this relationship further starting in the next section.

Regardless, in many cases, there is no exact notion of misclassification costs, meaning that there is no corresponding guidance as to which minority class training prior to use in practice [Mal03]. Moreover, in many domains, the exact misclassi-

fication costs may change over time [DH06], [PFK98]. While [WP03] shows there may be guidelines (depending on the evaluation metric being used) for choosing a good minority class training prior, the best minority class training prior varies from dataset to dataset and must be determined experimentally ([WP03], [EJJ04]). Thus, it is desirable for a resampling method to be robust with respect to the resampled minority class training prior.

In our experiments, random undersampling is particularly sensitive to choosing the correct resampled minority class training prior. In all six of our datasets, there is a very clear degradation in f-measure when the minority class training prior is either too low or too high regardless of whether C is tuned. Thus, while the best f-measure obtained using random undersampling is often competitive with results obtained from generative oversampling, the f-measure produced using random undersampling is extremely dependent on choosing the resampled minority class training prior.

Thus, generative resampling performs well compared to random oversampling, SMOTE, and random undersampling with respect to f-measure as well as with respect to robustness to minority class training prior. In contrast, SMOTE and random oversampling often cannot improve over using the natural prior (i.e., no resampling), particularly when C is tuned. Results with random undersampling depend heavily on choosing the appropriate minority class training prior, which may be difficult in practice.

3.3 Relationship Between Resampling and Cost-sensitive Learning

In [Elk01], a direct connection between cost-sensitive learning and resampling is made. The author shows that, theoretically, one can resample points at a spe-

cific rate in order to accomplish cost-sensitive learning. In section 4.1 of [Elk01], the author describes the effect of resampling on Bayesian classifiers. In particular, the author claims that resampling only changes the estimates of the prior probabilities. Thus, an equivalent model can be trained either through resampling or a cost-sensitive Bayesian model. In this section, we further examine the assumptions required for this equivalence to hold.

In this section, we examine the relationship between cost-sensitive learning and two oversampling methods: random oversampling and generative oversampling. We compare performance both theoretically and empirically using a variety of classifiers and data with different characteristics. In particular, we compare low versus high-dimensional data, and we compare Bayesian classifiers and support vector machines, both of which are very popular and widely used machine learning algorithms. Since there is already an abundance of empirical studies comparing resampling techniques and cost-sensitive learning, the emphasis of this section is to examine oversampling and its relationship with cost-sensitive learning from a theoretical perspective and to analyze the reasons for differences in empirical performance.

For low-dimensional data, assuming a Gaussian event model for the naive Bayes classifier, we show that random oversampling and generative oversampling theoretically increase the variance of the estimated sample mean compared to learning from the original sample (as done in cost-sensitive naive Bayes). Empirically, using generative oversampling and random oversampling seem to have minimal effect on Gaussian naive Bayes beyond adjusting the learned priors. This result implies that there is no significant advantage for resampling in this context. In contrast, for high-dimensional data, assuming a multinomial event model for the naive Bayes classifier, random oversampling and generative oversampling change not only the estimated priors, but also the parameter estimates of the multinomial distribution

modeling the resampled class. This conclusion is supported both theoretically and empirically. The theoretical analysis shows that oversampling and cost-sensitive learning are expected to perform differently in this context. Empirically, we demonstrate that oversampling outperforms cost-sensitive learning in terms of producing a better classifier.

Finally, we expand on the empirical results in the previous section where text classification was used with SVMs. In particular, we compare the results of resampling with cost-sensitive SVMs. We show empirically that generative oversampling used with linear SVMs provide the best results, beating any other combination of classifier and resampling/cost-sensitive method that we tested on our benchmark text datasets.

Empirically, there seems to be no clear winner as to which resampling technique to use or whether cost-sensitive learning outperforms resampling [Mal03] [MZW05] [WMZ07] [HKN07]. Many studies have been published, but there is no consensus on which approach is generally superior. Instead, there is ample empirical evidence that the best resampling method to use is dependent on the classifier [HKN07]. Since there is already an abundance of empirical studies comparing resampling techniques and cost-sensitive learning, the emphasis of this section is to examine oversampling and its relationship with cost-sensitive learning from a theoretical perspective and to analyze the reasons for differences in empirical performance.

3.3.1 A Theoretical Analysis of Oversampling Versus Cost-sensitive Learning

In this section, we study the effects of random and generative oversampling on Bayesian classification and the relationship to a cost-sensitive learning approach. We begin with a brief review of Bayesian classification and discuss necessary background. We then examine two cases and analyze differences in the estimates of the

parameters that must be calculated in each case to estimate the probability distributions being used to model the naive Bayes likelihoods. When multinomial naive Bayes is used, however, there is a significant difference. In this case, we show that the parameter estimates one obtains after either random or generative oversampling differ significantly from the parameter estimates used for cost-sensitive learning.

Bayesian Classification

Suppose one is solving a two-class problem using a Bayesian classifier. Let us denote the estimated conditional probability that some (possibly multi-dimensional) data point $\mathbf{x}$ is from the positive class y_+ given $\mathbf{x}$ as $\hat{P}(Y = y_+|\mathbf{X} = \mathbf{x})$ and the estimated conditional probability that $\mathbf{x}$ is from the negative class y_- given $\mathbf{x}$ as $\hat{P}(Y = y_-|\mathbf{X} = \mathbf{x})$. According to Bayes rule:

$$\hat{P}(Y = y_+|\mathbf{X} = \mathbf{x}) = \frac{\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+)\hat{P}(Y = y_+)}{\hat{P}(\mathbf{X} = \mathbf{x})} \tag{3.2}$$

and

$$\hat{P}(Y = y_-|\mathbf{X} = \mathbf{x}) = \frac{\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-)\hat{P}(Y = y_-)}{\hat{P}(\mathbf{X} = \mathbf{x})} \tag{3.3}$$

where $\hat{P}(Y = y_+)$ and $\hat{P}(Y = y_-)$ are the class priors, and $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+)$, $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-)$, $\hat{P}(Y = y_+)$, and $\hat{P}(Y = y_-)$ are estimated from the training set. $\hat{\theta}_+$ and $\hat{\theta}_-$ are the estimates of the parameters of the probability distributions being used to model the likelihoods $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+)$ and $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-)$. The decision rule is to assign $\mathbf{x}$ to y_+ if the posterior probability $\hat{P}(Y = y_+|\mathbf{X} = \mathbf{x})$ is greater than or equal to $\hat{P}(Y = y_-|\mathbf{X} = \mathbf{x})$. This is equivalent to classifying $\mathbf{x}$ as y_+ if

$$\frac{\hat{P}(Y = y_+|\mathbf{X} = \mathbf{x})}{\hat{P}(Y = y_-|\mathbf{X} = \mathbf{x})} \geq 1 \tag{3.4}$$

More generally, one can adjust the decision boundary by comparing the ratio of the two posterior probabilities to some constant. That is, one can adjust the decision boundary by assigning $\mathbf{x}$ to y_+ if

$$\frac{\hat{P}(Y = y_+|\mathbf{X} = \mathbf{x})}{\hat{P}(Y = y_-|\mathbf{X} = \mathbf{x})} \geq \alpha \tag{3.5}$$

where α is used to denote some constant. For example, one may use a particular value of α if one has known misclassification costs[Elk01]. If one were to use a cost-sensitive version of Bayesian classification, α is based on the cost c_+ of misclassifying a positive class point and the cost c_- of misclassifying a negative class point. In this case, $\alpha = c_-/c_+$, based on the cost-sensitive decision rule given in section 2.1.2.

One can also adjust the learned decision boundary by resampling (i.e., adding or removing points from the training set) which changes the estimated priors of the classes. Let $\hat{P}^{(rs)}(Y = y_+)$ and $\hat{P}^{(rs)}(Y = y_-)$ denote the estimated class priors after resampling (regardless of what resampling method has been used). Let $\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ and $\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})$ be estimated from the resampled training set. If $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ and $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})$, then the effect of using $\alpha \neq 1$ and the effect of adjusting the priors by resampling can be made exactly equivalent. That is, if resampling only changes the learned priors, then resampling at a specific rate corresponding to $\alpha = c_-/c_+$ is equivalent to cost-sensitive learning. In particular, one can show that if $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ and $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})$, then resampling to adjust the priors to correspond to $\alpha = c_-/c_+$ can be accomplished if

$$\alpha = \frac{c_-}{c_+} = \frac{\hat{P}^{(rs)}(Y = y_-)\hat{P}(Y = y_+)}{\hat{P}^{(rs)}(Y = y_+)\hat{P}(Y = y_-)} \tag{3.6}$$

However, in practice, this equivalency may not be exact since resampling may do more than simply adjust the class priors. That is, the assumption that $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ and $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})$ may be invalid because the estimated parameters with and without resampling may change.

In the remainder of this section, we will theoretically examine the effect of two resampling techniques (random oversampling and generative oversampling) on probability estimation and Bayesian learning. In particular, we will examine the difference between learning from the resampled set and learning from the original training set when Gaussian and multinomial distributions are chosen to model the resampled class.

We will assume without loss of generality that the positive class is being oversampled. In this case, since points are neither added nor removed from the negative class, $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})$. Thus, we will examine how $\hat{\theta}_+$ may differ from $\hat{\theta}_+^{(rs)}$ due to oversampling. Since we will only be discussing the positive class, we will omit the + subscripts when it is obvious that we are referring to parameters estimated on the positive class.

In addition, we will also use the notation $\hat{\theta}^{(r)}$ and $\hat{\theta}^{(g)}$ to refer to the parameters estimated after random oversampling and generative oversampling, respectively, when such a distinction needs to be made, while $\hat{\theta}^{(rs)}$ will continue to refer to parameter estimates after either resampling technique has been used, and $\hat{\theta}$ will continue to refer to parameters estimated from the original training set when no resampling has occurred.

Effect of Oversampling on Gaussian Naive Bayes

In this section, we examine the effect of oversampling when $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+)$ and $\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ are modeled by Gaussian distributions. In Gaus-

sian naive Bayes, each feature is modeled by an independent Gaussian distribution. Thus, $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+) = \prod_{i=1}^{d} N(x_i|\mu_{+,i}, \sigma_{+,i})$ where d is the number of dimensions, $\mu_{+,i}$ and $\sigma_{+,i}$ are the mean and standard deviation estimated for the ith dimension of the positive class, and $N(x_i|\mu_{+,i}, \sigma_{+,i})$ is the probability that a normal distribution with parameters $\mu_{+,i}$ and $\sigma_{+,i}$ generated x_i. Since we are only discussing oversampling the positive class, we will drop the + subscripts and simply refer to the parameters as $\hat{\theta}$, μ_i, and σ_i.

For the sake of simplicity, we will limit our discussion to one-dimensional Gaussian distributions. However, since the parameters of each dimension are estimated independently in a naive Bayes classifier, our analysis can be extended to multiple dimensions if the features are indeed independent. Analysis when features are correlated will be left to future work. In the one-dimensional case, $\hat{\theta}$ corresponds to a single sample mean and sample standard deviation estimated from the positive class points in the original training set, while $\hat{\theta}^{(rs)}$ corresponds to the sample mean and sample standard deviation estimated after resampling.

We will first examine the theoretical effect of oversampling on estimating $\hat{\theta}$, $\hat{\theta}^{(r)}$, and $\hat{\theta}^{(g)}$. For the sake of brevity, we limit our discussion of these parameter estimates to the expected value and variance of the sample mean. In particular, we show that the expected value of the sample mean is always the same regardless of whether no resampling, random oversampling, or generative oversampling is applied.

Let the set of n points in the positive class be denoted as $X_1, \ldots, X_n$. These points are an iid set of random variables from a normal distribution with true mean μ and true variance σ^2. Both μ and σ^2 are unknown and must be estimated as parameters $\hat{\theta}$.

The sample mean estimated from the original training set will be denoted as $\bar{X}$ while the sample mean for the training set created after resampling will be denoted as either $\bar{X}_*^{(r)}$ for random oversampling or $\bar{X}_*^{(g)}$ for generative oversampling. Note

that, in the Appendix, we make a differentiation between the sample mean estimated only on the newly resampled points (denoted as $\bar{X}^{(r)}$ for random oversampling) versus the sample mean estimated for a training set comprised of both the resampled points and the original points (denoted as $\bar{X}_*^{(r)}$). Here, however, we will simply present results for the "pooled" training set consisting of both the resampled points and the original points.

Consider the sample mean of the original sample, $\bar{X}$. The expected value and variance of the sample mean is

$$E[\bar{X}] = \mu \tag{3.7}$$

$$\text{Var}[\bar{X}] = \frac{\sigma^2}{n} \tag{3.8}$$

In the next sections, we will find the expected value and variance of the sample mean of the points generated by random oversampling and generative oversampling. Derivations of these equations can be found in the Appendix.

Random Oversampling

Consider a random sample $X_1^{(r)}, \ldots, X_m^{(r)}$ produced through random oversampling. Thus, each $X_j^{(r)} = X_{K_j}$, where $K_1, \ldots, K_m$ are iid uniformly distributed random variables over the discrete set $\{1, \ldots, n\}$.

We now seek the mean and variance of $\bar{X}_*^{(r)}$.

For the mean, we have

$$E[\bar{X}_*^{(r)}] = \mu \tag{3.9}$$

and

$$\text{Var}[\bar{X}_*^{(r)}] = \left[1 + \frac{m(n-1)}{(n+m)^2}\right] \frac{\sigma^2}{n} \tag{3.10}$$

Thus, the expected value of the sample mean of the pooled training set is equal to the expected value of the sample mean without resampling. However, the variance of the estimated sample mean of the training set after resampling is greater.

Generative Oversampling

Now consider points $X_1^{(g)}, \ldots, X_m^{(g)}$ created via generative oversampling. These points are of the form

$$\mathbf{X}_j^{(g)} = \bar{\mathbf{X}} + s\mathbf{Z}_j, \tag{3.11}$$

where $X_j^{(g)}$ is the j^{th} point created by generative oversampling, s is the original estimated sample standard deviation, and $Z_1, \ldots, Z_m$ are iid $N(0,1)$ and independent of $X_1, \ldots, X_n$ as well.

The expected value of the mean $\bar{X}_*^{(g)}$ is

$$E[\bar{X}_*^{(g)}] = \mu \tag{3.12}$$

while the variance of the sample mean is

$$\text{Var}[\bar{X}_*^{(g)}] = \left[1 + \frac{mn}{(n+m)^2}\right] \frac{\sigma^2}{n} \tag{3.13}$$

Thus, like random oversampling, the sample mean estimated from the resampled points created via generative oversampling has, on average, the same value as the sample mean estimated from the original points, but with greater variance.

Comparison to cost-sensitive learning

A cost-sensitive naive Bayes classifier uses the parameter estimates $\hat{\theta}$ from the original set of points. Thus, in expectation, the estimated mean for a Gaussian naive Bayes classifier will be the same, regardless of whether random oversampling, generative oversampling, or cost-sensitive learning are used. When one resamples, one incurs additional overhead in terms of time required to create additional samples,

memory needed to store the additional samples, and additional time required to train on the resampled points. Cost-sensitive naive Bayes is therefore preferable over resampling when a Gaussian distribution is assumed. We will support this claim empirically in section 3.3.2.

Effects of Oversampling for Multinomial Naive Bayes

In multinomial naive Bayes (see [MN98] for an introduction to multinomial naive Bayes), there is a set of d possible features, and the probability that each feature will occur needs to be estimated. For example, in the case of text classification, each feature is a word in the vocabulary, and one needs to estimate the probability that a particular word will occur. Thus, the parameter vector θ for a multinomial distribution is a d dimensional vector, where θ_k is the probability that the kth word in the vocabulary will occur.

Let F_i denote the number of times the ith word occurs in the positive class in the training set, and let $F_i^{(r)}$ represent the number of times that a word occurs in only the randomly oversampled points (we will use $F_i^{(g)}$ when discussing generative oversampling). In addition, let n represent the number of words that occur in the positive class in the training set, and let m represent the number of words that occur in the resampled points.

For the case where there are no resampled points in the training set, the maximum likelihood estimator for the probability the kth word will occur is $\hat{\theta}_k = F_k/n$. Typically, the maximum likelihood estimator is not used because Laplace smoothing is often introduced (a standard practice when using multinomials for problems like text mining). With Laplace smoothing, the estimator becomes

$$\tilde{\theta}_k = \frac{F_k + 1}{\sum_{k'=1}^{d}(F_{k'} + 1)} = \frac{n\hat{\theta}_k + 1}{n + d}, \tag{3.14}$$

Note that we will use the notation $\hat{\theta}$ to describe parameter estimates when

Laplace smoothing has not been used and $\tilde{\theta}$ to indicate parameter estimates when Laplace smoothing has been used.

After random oversampling, we find that the parameter estimates learned from the resampled points and original training set if Laplace smoothing is used is:

$$E\left[\tilde{\theta}_k^{(r)}\right] = \frac{(n+m)\theta_k + 1}{n+m+d} \tag{3.15}$$

and

$$\text{Var}\left[\tilde{\theta}_k^{(r)}\right] = \left[1 + \frac{m(n-2d-1) - d(2n+d)}{(n+m+d)^2}\right] \frac{\theta_k(1-\theta_k)}{n}. \tag{3.16}$$

Thus, provided $m < d(2n+d)/(n-2d-1)$, the variance of the pooled, smoothed estimates will be smaller than that of the original sample.

In generative oversampling, one generates points based on an assumed probability distribution. If one uses a multinomial model to generate the points, the parameters one uses in the initial estimation can be either $\hat{\theta}$ (i.e., without Laplace smoothing) or $\tilde{\theta}$ (i.e., with Laplace smoothing). When using generative oversampling with multinomial naive Bayes, there are two places where Laplace smoothing can possibly be used: when performing the initial parameter estimates for generative oversampling and when performing the parameter estimates for multinomial naive Bayes. In our experiments, we always use Laplace smoothing when estimating parameters for multinomial naive Bayes. The question of whether to use Laplace smoothing will therefore always refer to the initial parameter estimates in generative oversampling.

As shown in the Appendix, if one uses $\hat{\theta}$ for their initial parameter estimates in generative oversampling, then $E\left[\tilde{\theta}_k^{(g)}\right] = E\left[\tilde{\theta}_k^{(r)}\right]$ and $\text{Var}\left[\tilde{\theta}_k^{(g)}\right] = \text{Var}\left[\tilde{\theta}_k^{(r)}\right]$. If one performs Laplace smoothing and uses $\tilde{\theta}$ in the initial parameter estimates used for generative oversampling, however, the parameter vector $\tilde{\theta}^{(g)}$ estimated after generative oversampling will be different from the parameter vector $\tilde{\theta}^{(r)}$ estimated

after random oversampling or the parameter vector $\tilde{\theta}$ used in cost-sensitive learning. Our empirical results show that the relative performance of using either $\hat{\theta}$ or $\tilde{\theta}$ when estimating the initial parameters used in generative oversampling depends on which classifier is used.

Since a cost-sensitive naive Bayes classifier uses the parameter estimates $\tilde{\theta}$ from the original set of points, it is clear that there will be a difference, in expectation, between the parameters estimated via random oversampling, generative oversampling, and cost-sensitive learning. We will see empirically that resampling produces better classifiers than cost-sensitive learning. Thus, even though resampling incurs additional overhead in terms of time and memory, the improvement in classification may justify this additional effort.

3.3.2 Empirical Comparison of Resampling and Cost-Sensitive Learning

In this section, we will provide empirical support for our analysis in section 3.3.1. We will show that, as predicted, there is minimal empirical difference between random oversampling, generative oversampling, and cost-sensitive learning when Gaussian naive Bayes is used as the classifier. In contrast, when dealing with high-dimensional text datasets where a multinomial model is more suitable, there is a difference between random oversampling, generative oversampling, and cost-sensitive learning. The magnitude of the difference with regards to generative oversampling is related to whether Laplace smoothing is used to build the model used to generate artificial points.

Explaining Empirical Differences Between Resampling and Cost-sensitive Learning

Our experiments compare the results of classifiers learned after resampling against a cost-sensitive classifier that estimates its parameters from the original training set. In this section, we will describe why comparing naive Bayes after resampling with cost-sensitive naive Bayes can answer the question of whether the benefits of resampling are limited to merely evening out the imbalance of the class priors, or if additional effects (from changing the estimates of the likelihoods) are responsible.

Oversampling the positive class has two possible effects on a Bayesian classifier: (1) it changes the estimated priors $\hat{P}(Y = y_+)$ and $\hat{P}(Y = y_-)$ to $\hat{P}^{(rs)}(Y = y_+)$ and $\hat{P}^{(rs)}(Y = y_-)$, and (2) it may or may not change the parameter estimate $\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ such that $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+) \neq \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$.

The decision rule of the Bayesian classifier after resampling is to assign a point x to the positive class if

$$\frac{\hat{P}^{(rs)}(Y = y_+|\mathbf{X} = \mathbf{x})}{\hat{P}^{(rs)}(Y = y_-|\mathbf{X} = \mathbf{x})} = \frac{\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})\hat{P}^{(rs)}(Y = y_+)}{\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})\hat{P}^{(rs)}(Y = y_-)} \geq 1 \qquad (3.17)$$

As described in the previous section, if $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ and $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})$, then the only effect of resampling is to change the learned class priors. Under this (possibly incorrect) assumption,

$$\frac{\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})\hat{P}^{(rs)}(Y = y_+)}{\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})\hat{P}^{(rs)}(Y = y_-)} = \frac{\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+)\hat{P}^{(rs)}(Y = y_+)}{\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-)\hat{P}^{(rs)}(Y = y_-)} \qquad (3.18)$$

This is the same as adjusting the decision rule learned on the original train-

ing set by setting $\alpha = \frac{\hat{P}^{(rs)}(Y=y_-)\hat{P}(Y=y_+)}{\hat{P}^{(rs)}(Y=y_+)\hat{P}(Y=y_-)}$ and assigning $\mathbf{x}$ to the positive class if $\frac{\hat{P}(Y=y_+|\mathbf{X}=\mathbf{x})}{\hat{P}(Y=y_-|\mathbf{X}=\mathbf{x})} \geq \alpha$. One can do this by training a cost-sensitive naive Bayes classifier with $\alpha = \frac{c_-}{c_+} = \frac{\hat{P}^{(rs)}(Y=y_-)\hat{P}(Y=y_+)}{\hat{P}^{(rs)}(Y=y_+)\hat{P}(Y=y_-)}$.

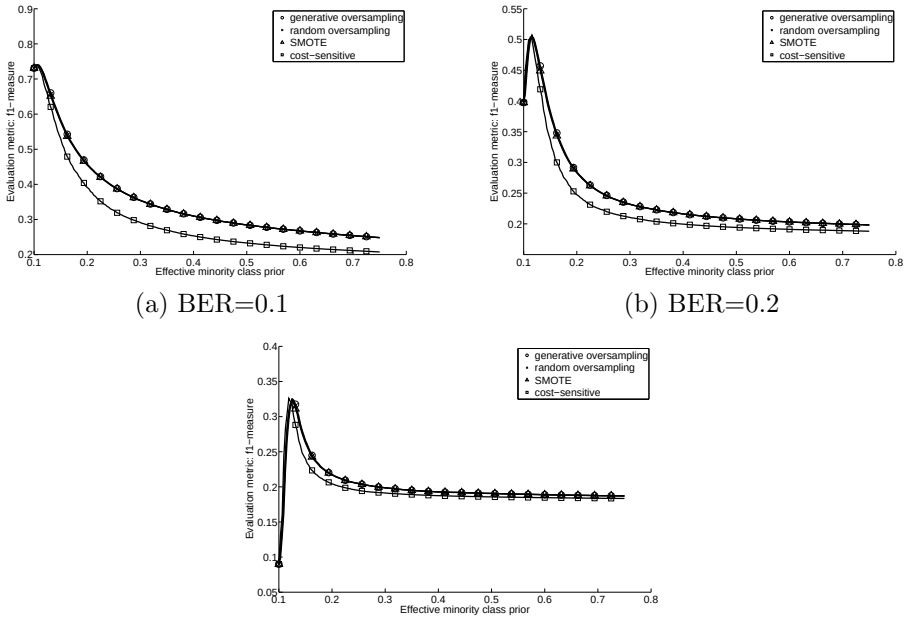
Thus, one can duplicate the beneficial effect of evening out the class priors via resampling if $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+) = \hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+^{(rs)})$ and $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_-,\hat{\theta}_-) = \hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_-,\hat{\theta}_-^{(rs)})$ by using a cost-sensitive naive Bayes classifier where $\frac{c_-}{c_+} = \frac{\hat{P}^{(rs)}(Y=y_-)\hat{P}(Y=y_+)}{\hat{P}^{(rs)}(Y=y_+)\hat{P}(Y=y_-)}$. Any empirical difference observed between a naive Bayes classifier after resampling and a cost-sensitive naive Bayes classifier with the appropriate values of c_- and c_+ is therefore attributable to the fact it is incorrect to assume that the estimated parameters modeling our probability distributions are equal before and after resampling.

Therefore, we can examine whether $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+) = \hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+^{(rs)})$ by comparing a naive Bayes classifier that uses resampling and an equivalent cost-sensitive naive Bayes classifier where $\frac{c_-}{c_+} = \frac{\hat{P}^{(rs)}(Y=y_-)\hat{P}(Y=y_+)}{\hat{P}^{(rs)}(Y=y_+)\hat{P}(Y=y_-)}$. Such a comparison allows us to isolate and study only the part of resampling that could cause $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+) \neq \hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+^{(rs)})$. We perform this comparison in the upcoming sections 3.3.2 and 3.3.2.

Naive Bayes Comparisons on Low-dimensional Gaussian Data

In this section, we will provide some simple examples of classifying low-dimensional data with a Gaussian naive Bayes classifier to support the theory presented in 3.3.1.

We use f1-measure, a natural evaluation metric in information retrieval for high-dimensional datasets, as our evaluation metric. As mentioned, f1-measure is the harmonic mean of two other evaluation metrics, precision and recall. Precision = $n_{tp}/(n_{tp}+n_{fp})$ and recall = $n_{tp}/(n_{tp}+n_{fn})$, where n_{tp} is the number of true positives, n_{fp} is the number of false positives, and n_{fn} is the number of false negatives. F1-measure ranges from 0 to 1, with 1 being the best possible f1-measure achievable



(a) BER=0.1

(b) BER=0.2

(c) BER=0.3

Figure 3.4: Results on artificial Gaussian data for naive Bayes

on the test set. F1-measure has some additional advantages over traditional ROC and AUC metrics when interpreting our experiments, as discussed in 3.3.2. In order to keep our results consistent, we use f1-measure for both the low-dimensional and high-dimensional experiments.

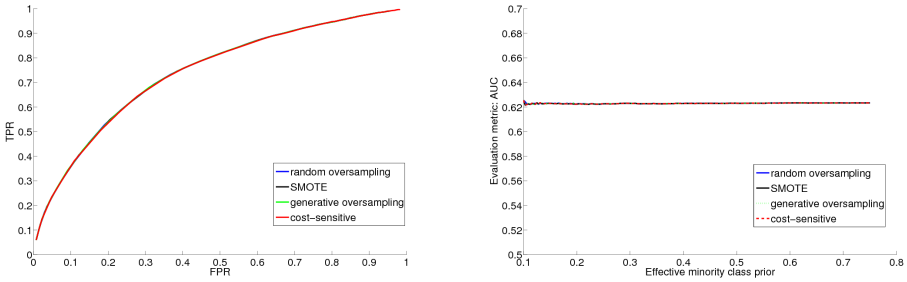
To illustrate that there is minimal benefit in using either random oversampling, generative oversampling, or cost-sensitive learning when a Gaussian naive Bayes classifier is used, we present two sets of experiments.

Gaussian Naive Bayes on artificial, low-dimensional data

The first set of experiments utilizes an artificially generated dataset consisting of two classes drawn from two 1-d Gaussians with true variance equal to 1. The location of the means of the two Gaussians is controlled such that a specific optimal Bayes error rate could potentially be achieved if the points in the test data were sampled equally from the two distributions (the Bayes error rate is defined as the lowest

theoretical error rate achievable if we knew the true values of the parameters in our mixture of Gaussians [DHS01]). We vary the optimal Bayes error rate (denoted as BER in figure 3.4) between 0.1 and 0.3. In order to introduce imbalance, 90% of the training set consists of points in the negative class and 10% of the training set consists of points in the positive class. In our experiments, we varied the amount of training data between 100 and 300 points, but found that the results were consistent regardless of how much training data was used; here, we present results where there are 100 training points. The test set consists of 1000 points with the same priors as the training set. We average our results over 100 trials, where each trial includes creating a completely new data set.

When resampling, one has control over the value of the estimated priors $\hat{P}^{(rs)}(Y = y_+)$ and $\hat{P}^{(rs)}(Y = y_-)$. Since $\hat{P}^{(rs)}(Y = y_+) + \hat{P}^{(rs)}(Y = y_-) = 1$, controlling $\hat{P}^{(rs)}(Y = y_+)$ is sufficient to control both $\hat{P}^{(rs)}(Y = y_+)$ and $\hat{P}^{(rs)}(Y = y_-)$. In our experiments, we vary $\hat{P}^{(rs)}(Y = y_+)$ between the prior estimated without resampling $\hat{P}(Y = y_+) = 10\%$ and a maximum possible value of 75%. Note that when $\hat{P}^{(rs)}(Y = y_+) = \hat{P}(Y = y_+)$, no actual resampling has been performed (this corresponds to the left-most point on each graph plotting f1-measure where all performance curves converge). When running cost-sensitive learning, we control the misclassification costs c_- and c_+. In order to directly compare cost-sensitive learning against results for resampling, we use the term "effective minority class prior" in our graphs. That is, a particular value of the effective minority class prior means that: (1) when resampling is used, the resampled prior $\hat{P}^{(rs)}(Y = y_+)$ is equal to the effective minority class prior, and (2) when cost-sensitive learning is used, $\frac{c_-}{c_+} = \frac{\hat{P}^{(rs)}(Y=y_-)\hat{P}(Y=y_+)}{\hat{P}^{(rs)}(Y=y_+)\hat{P}(Y=y_-)}$, where $\hat{P}^{(rs)}(Y = y_+)$ is equal to the effective minority class prior. In interpreting our results, we simply look at our results on the test set across a range of resampling rates. Choosing a resampling rate that yields optimal performance is an unsolved problem. That is, there is no closed form



(a) Example ROC curves for artificial data (BER=0.3)

(b) Example AUC curves for artificial data (BER=0.3)

Figure 3.5: Example results using AUC and ROC as evaluation metrics; note that, using these evaluation metrics, it is difficult to determine whether the effective minority class prior has any effect on how well the classifier performs.

solution for determining the appropriate effective minority class prior to maximize a particular evaluation metric, so this becomes a model selection problem.

In figure 3.4, we plot the f1-measure versus different effective minority class priors for Gaussian naive Bayes after random oversampling, Gaussian naive Bayes after generative oversampling, Gaussian naive Bayes after SMOTE, and cost-sensitive naive Bayes. Interestingly, regardless of the separability of the two Gaussian distributions, the curves have similar characteristics. In particular, the best possible f1-measure obtained by each is almost exactly the same. That is, in practice, random oversampling, generative oversampling, and SMOTE seem to have little effect on Gaussian naive Bayes that cannot be accomplished via cost-sensitive learning. This supports the theory presented in section 3.3.1, which shows that, in expectation, the value of the sample mean estimated after random oversampling, generative oversampling, or cost-sensitive learning (which uses the original set of points) is equivalent.

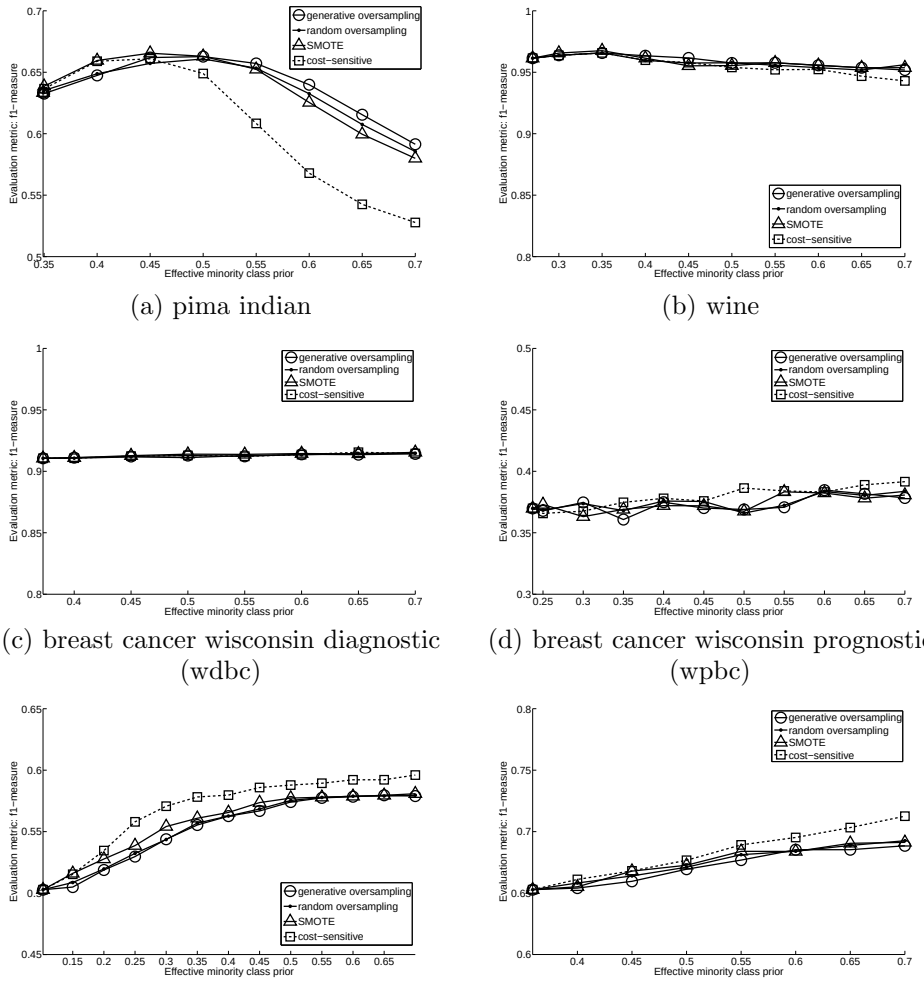
A note on ROC and AUC Figure 3.5 contains ROC curves and values for AUC for the same set of experiments presented in figure 3.4 where BER=0.3. When evaluating results on imbalanced datasets, ROC and AUC are often very useful. However, ROC and AUC can hide the effect that different resampling rates have on

the classifier. To fully examine the effect of resampling rate using ROC curves would require an unwieldy number of curves per graph, since each resampling rate for each method being used would require a separate ROC curve. Figure 3.5, which seems to contain only a single ROC curve, is an exception since, as theory predicts, each ROC curve produced by each resampling method and cost-sensitive Gaussian naive Bayes is essentially the same (thus retraced multiple times in the figure). Another problem is that to create a ROC curve, one uses several different thresholds for each point on the curve; cost-sensitive naive Bayes also uses different thresholds to produce results using different costs. Thus, the differences in performance across different costs are hidden on a single ROC curve for this type of classifier. AUC, which is based on ROC, aggregates results over several thresholds. Thus, the AUC for cost-sensitive naive Bayes will always be about the same (sans statistical variation in the training/test sets) regardless of the cost used, and is not particularly interesting. In fact, this is exactly what we see in figure 3.5, where the AUC remains essentially constant regardless of the effective minority class prior. The results in figure 3.5 can be extremely misleading, because it may lead one to conclude that different cost parameters or resampling rates have no effect on how well a classifier performs. In comparison, using f1-measure to plot un-integrated results corresponding to specific resampling rates and costs, clearly shows the importance of choosing an appropriate resampling rate or cost.

Gaussian Naive Bayes on real, low-dimensional data

To complement the experiments on Gaussian naive Bayes on artificial data, we also present some results on low-dimensional datasets from the UCI dataset repository to verify generalizations of the findings on real data. We have selected six datasets: pima indian, wine, breast cancer wisconsin diagnostic (wdbc), breast cancer wisconsin prognostic (wpbc)[4], page-blocks (using "non-text" versus "rest" as

[4]Note that the two breast cancer datasets are separate datasets in the UCI dataset repository, and not the same dataset used for two different tasks in our experiments.



(a) pima indian

(b) wine

(c) breast cancer wisconsin diagnostic (wdbc)

(d) breast cancer wisconsin prognostic (wpbc)

(e) page-blocks (non-text vs. rest)

(f) ionosphere

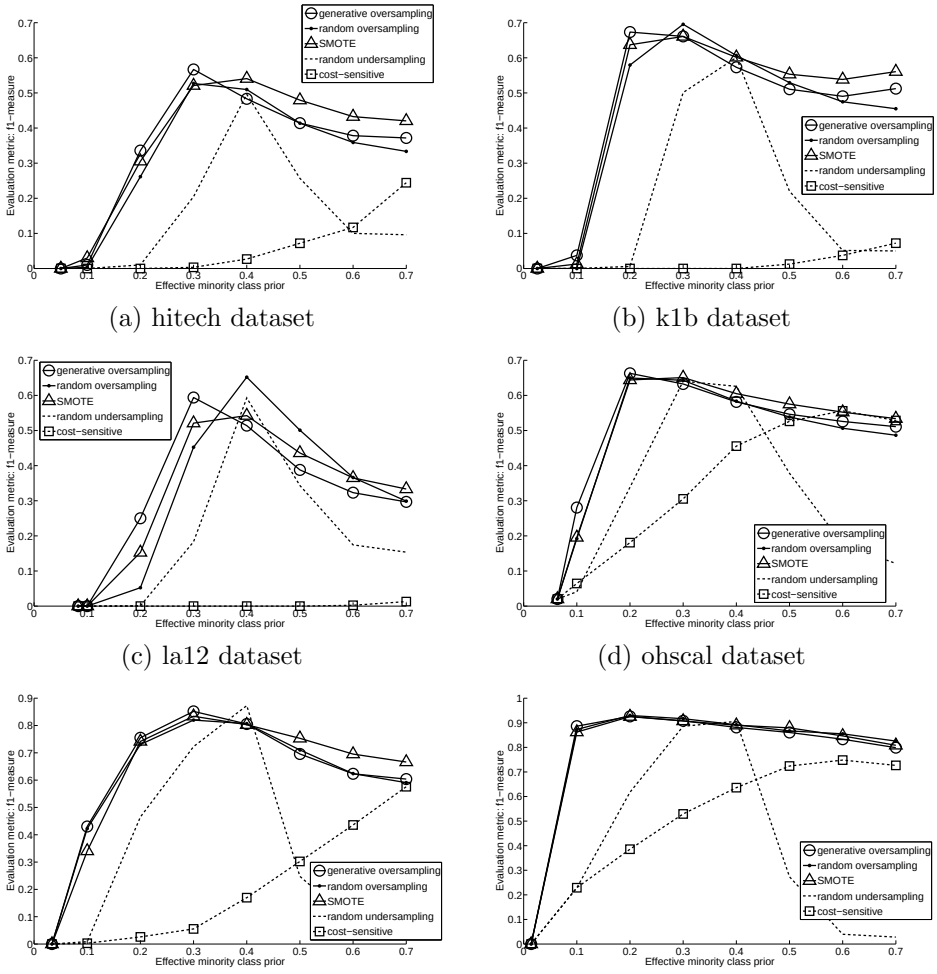
Figure 3.6: Results on real datasets from UCI dataset repository using gaussian naive Bayes

our binary class problem), and ionosphere. The features of all of the dimensions for wine and breast cancer wisconsin diagnostic (wdbc) pass the Kolmogorov-Smirnov test for normality for a p-value of 0.05; most of the features of the breast cancer wisconsin prognostic (wpbc) dataset also pass the Kolmogorov-Smirnov test for normality. In contrast, the majority of the features of the remaining datasets did not. We use both features that passed and did not pass the Kolmogorov-Smirnov test in our experiments, so the assumption that Gaussians can be used to model the various datasets does not hold very well on some of the datasets.

Our results are shown in figure 3.6. The results of these experiments support the same conclusion as before: there is little advantage to using either random oversampling, generative oversampling, or cost-sensitive learning when using Gaussian naive Bayes. For the sake of comparison, SMOTE is included in these datasets as well. While SMOTE has been shown to work well with other classifiers [CBHK02], it performs similarly to the other techniques when using Gaussian naive Bayes. Thus, given that it is much easier to use cost-sensitive learning instead of resampling and that there is no empirical advantage of using resampling, it is preferable to simply use a cost-sensitive version of naive Bayes when Gaussian distributions are used to model the data.

Multinomial Naive Bayes

In this section and the next, we examine the empirical effect of resampling on high-dimensional data. In particular, we will use text classification as an example domain. We first examine the effect of random and generative oversampling on multinomial naive Bayes [MN98], a classifier often used for text classification. Our experiments on text classification are more extensive than the experiments on low-dimensional data for two primary reasons: 1) additional results more fully illustrate the empirical differences between the different resampling methods, and 2) empirical studies



(a) hitech dataset

(b) k1b dataset

(c) la12 dataset

(d) ohscal dataset

(e) reviews dataset

(f) sports dataset

Figure 3.7: Results on text datasets using multinomial naive Bayes

comparing resampling methods and/or cost-sensitive learning typically focus on low-dimensional data, so there are less published results available for high-dimensional data.

We compare the effect of random oversampling, generative oversampling, and cost-sensitive learning on multinomial naive Bayes using the same six text datasets and preprocessing steps used in our SVM experiments in section 3.2.5.

As in the experiments with Gaussian data, we control the effective minority class prior either through resampling or cost-sensitive learning. Again, we vary the effective minority class prior between the prior estimated without resampling $\hat{P}(Y = y_+)$ and 70% (note that the prior before resampling varies for each dataset, but is always less than 10%).

The results of our experiments on multinomial naive Bayes are shown in figure 3.7. The results indicate that resampling improves the resulting f1-measure when compared to classification without resampling (i.e., the left-most point in each graph). The results also indicate that there is a significant difference between the best possible f1-measure obtained from resampling (across all resampling rates) and the best possible f1-measure obtained through cost-sensitive learning (across all tested costs). In particular, the best possible f1-measure obtained after oversampling (regardless of which oversampling method we tested) is always better than cost-sensitive learning. In some cases, this value is much higher than the best possible f1-measure obtained via cost-sensitive learning.

Both random oversampling, generative oversampling,[5] and SMOTE produce comparable f1-measure curves. These results indicate that, in practice, one can produce a classifier with better performance by oversampling instead of adjusting the decision boundary using cost-sensitive learning.

[5]Here, we use generative oversampling as described in the Appendix where $\hat{\theta}$ (i.e., without smoothing during parameter estimation) is used instead of $\tilde{\theta}$ to generate points; we will see in section 3.3.2 why this distinction is important.

SVMs

In this section, we empirically test the effect of resampling on linear SVMs in the domain of text classification and compare performance against cost-sensitive SVMs.[6] These experiments are an extension of those presented in section 3.2.5. In short, we now add cost-sensitive learning to the results presented in section 3.2.5. Empirical set-up is the same as in section 3.2.5, and is repeated here for the sake of convenience.

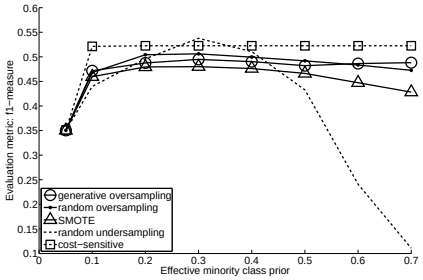
In our experiments, we use an SVM with a linear kernel, which has been shown to work well in text classification [Yan99]. Specifically, we use the SVM-light implementation by Joachims [Joa98]. Note that, for SVMs, there is an additional parameter used to control the trade-off between the margin and training errors. In SVM-light this is the parameter C.[7] Adjusting C can affect the location of the separating hyperplane. In SVM-light, one can either specify C or use a default value of C estimated from the data. We run experiments using both settings, and the results are presented separately in figures 3.8 and 3.9.

Figure 3.8 plots our results comparing generative oversampling, random oversampling, SMOTE, random undersampling, and cost-sensitive SVMs on each of the six datasets when all default parameters for linear SVMs are used. The plots show f1-measure as a function of effective minority class prior, as presented for the naive Bayes results.

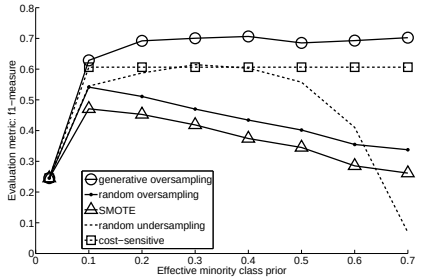
Figure 3.9 shows the results when the tradeoff C between margin size and number of training errors is tuned via a validation set. When specifying C, we perform a search for the best value of C by using a validation set; we further split each training set into a smaller training set (70% of initial training set) and validation

[6] All SVM results presented use generative oversampling for multinomials with smoothing during parameter estimation ($\tilde{\theta}$) except for figure 3.10, where we present results for generative oversampling with both $\tilde{\theta}$ and $\hat{\theta}$. Generative oversampling for multinomials without smoothing (i.e., when $\hat{\theta}$ is used when estimating the parameters for generative oversampling) performs poorly for SVMs as shown at the end of this section.

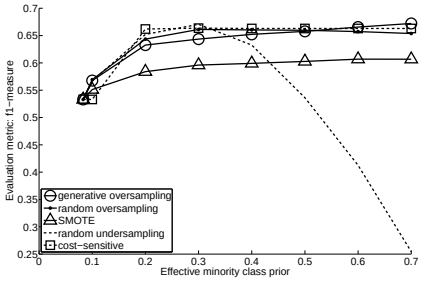
[7] Not to be confused with a cost matrix $\mathbf{C}$.



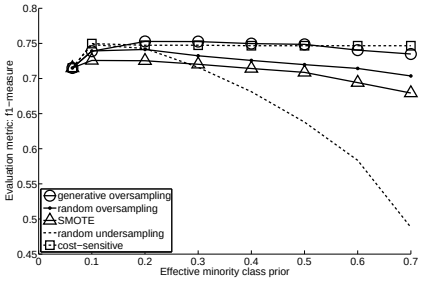
(a) hitech dataset



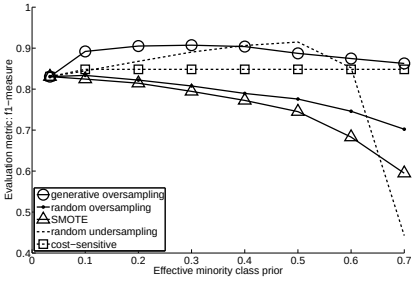
(b) k1b dataset



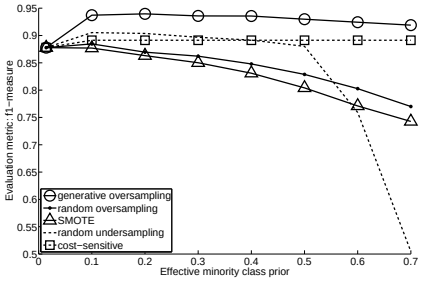
(c) la12 dataset



(d) ohscal dataset

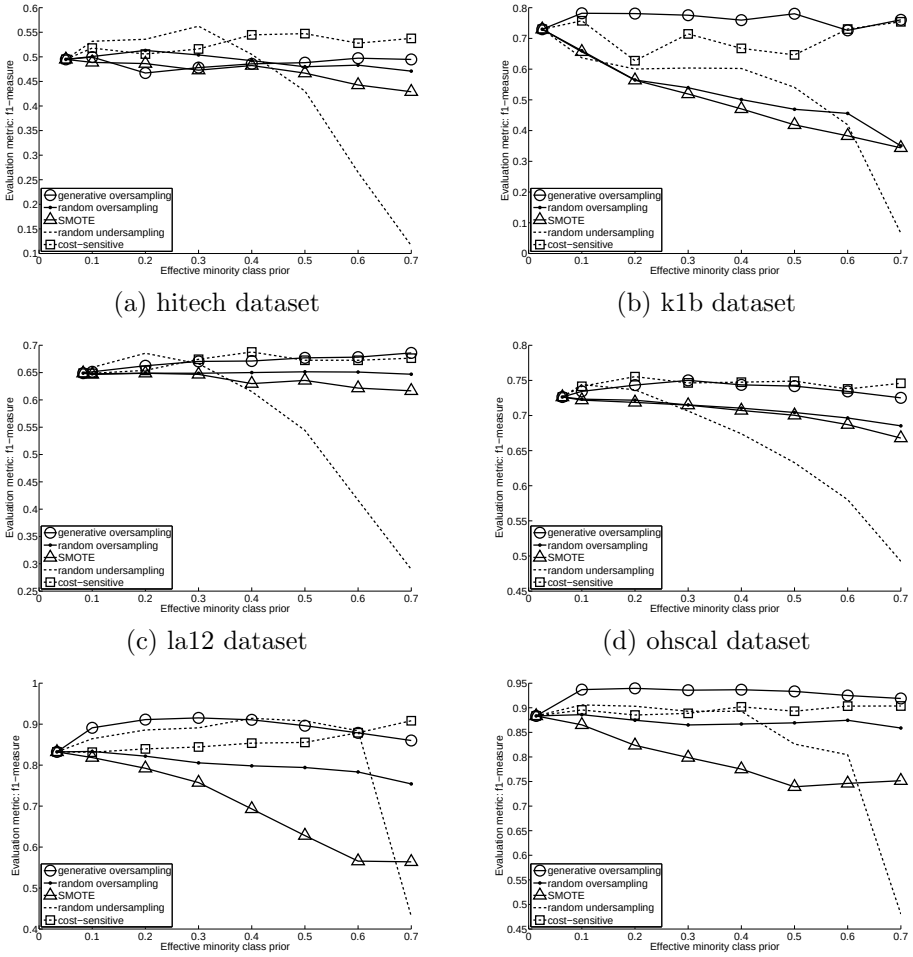


(e) reviews dataset



(f) sports dataset

Figure 3.8: Results on text datasets using SVMs without tuning C



(a) hitech dataset

(b) k1b dataset

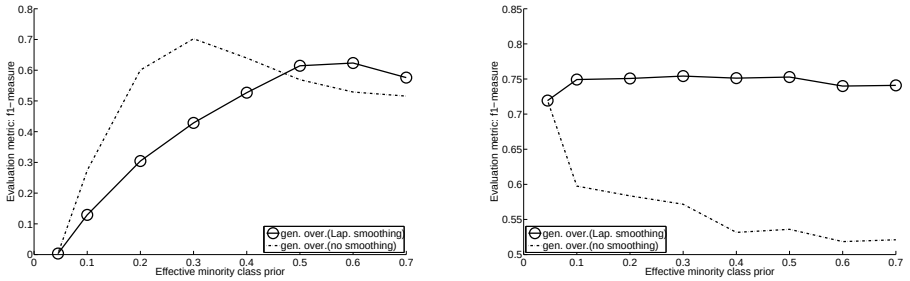
(c) la12 dataset

(d) ohscal dataset

(e) reviews dataset

(f) sports dataset

Figure 3.9: Results on text datasets using SVMs while tuning C



(a) averaged mult. NB (b) averaged SVMs

Figure 3.10: Average results on text datasets for multinomial naive Bayes and SVMs (where C is tuned) after running generative oversampling with different levels of smoothing

set (the remaining 30% of the training set) and search for the best value of C between 2^{-6} and 2^{6}. Note that this tuning is done separately for every experiment (i.e., once for every combination of dataset, training set split, resampled minority class prior, and resampling method).

In both sets of experiments, generative oversampling performs well compared to random oversampling, SMOTE, random undersampling, and cost-sensitive SVMs. Results comparing the resampling methods were previously discussed. Cost-sensitive SVMs perform quite well and are as robust to choice of cost parameter as generative oversampling is to choosing how much to resample, but on average, generative oversampling produces a higher f1-measure.

If one runs SVMs with resampled points created via generative oversampling with Laplace smoothing during parameter estimation (i.e., if one uses $\tilde{\theta}$), then generative oversampling potentially increases the size of the convex hull surrounding the minority class by producing artificial data points that occur both inside and outside of the original convex hull inscribing the minority class points in the training set. Figure 3.10 contains averaged results where we compare generative oversampling on SVMs using either $\tilde{\theta}$ or $\hat{\theta}$ (i.e., generative oversampling with and without Laplace smoothing). As one can see, the results when using $\hat{\theta}$ are much worse. When $\hat{\theta}$ is used, generative oversampling no longer effectively creates points outside of the

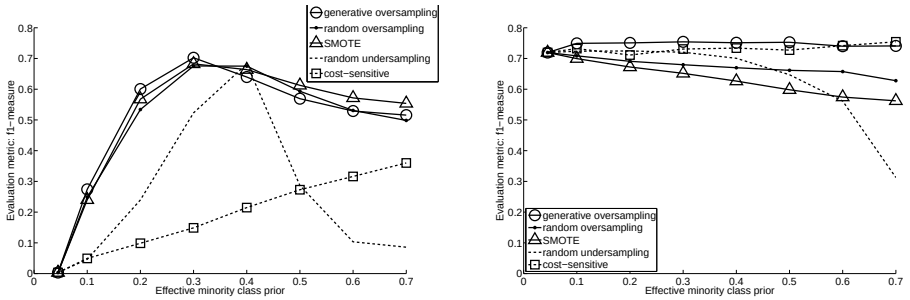
original convex hull. Thus, generative oversampling with $\tilde{\theta}$ complements the SVMs well by increasing the size of the minority class convex hull.

Note that, in our multinomial naive Bayes experiments, we found that using $\hat{\theta}$ was always empirically superior to using $\tilde{\theta}$, while for SVMs, we found that using $\tilde{\theta}$ resulted in better empirical performance (see figure 3.10 for graphs of each case). That is, even the same resampling method in the same problem domain interacts very differently with different classifiers.

Finally, we observe that all of the results that work well with SVMs move the separating hyperplane in some fashion, either by 1) changing the shape of the convex hulls inscribing either the minority class (generative oversampling) or majority class (random undersampling) or 2) changing the location of the separating hyperplane by controlling the tradeoff between margin and number of empirical mistakes in the training set (tuning the parameter C or using cost-sensitive SVMs). Our analysis supports the conclusion that random oversampling and SMOTE, which work well for the multinomial naive Bayes classifier, have minimal effect on SVMs since neither are effective at changing the shape of the minority class convex hull. In fact, if one tunes the parameter C, then neither random oversampling nor SMOTE are useful.

3.3.3 Discussion

In summary, we have presented experiments which can be divided into two cases: in the first case, there is no advantage to performing resampling as opposed to simply performing cost-sensitive learning. Examples of this were presented for artificial and real low-dimensional datasets for a Gaussian naive Bayes classifier. In the second case, there is a clear difference in performing resampling as opposed to cost-sensitive learning due to various effects caused by the resampling methods. Several examples of this effect were presented using multinomial naive Bayes and linear SVMs on high-dimensional text datasets. Empirically, we showed that, for both naive Bayes



(a) averaged mult. NB results (b) averaged SVM results

Figure 3.11: Average results on text datasets for multinomial naive Bayes and SVMs (where C is tuned)

and SVMs on text classification, resampling resulted in a better classifier than cost-sensitive learning.

Averaged results for the text datasets used in these experiments are presented in figure 3.11. When trying to achieve the optimal f1-measure on these text datasets, figure 3.11 shows that the best approach is to use linear SVMs with generative oversampling.

Our results support the conclusion that the best resampling method (or whether cost-sensitive learning outperforms resampling) is dependent on the dataset and classifier being used. This has been seen empirically in other papers such as [HKN07]. Our analysis provides some insight for why these differences occur.

3.4 Summary

In this chapter, we have studied methods for handling the imbalanced dataset problem, a problem that exhibits both relative and absolute data scarcity. We first study resampling by introducing the generative oversampling algorithm, a resampling algorithm that creates artificial data points from a probability distribution learned from the minority class. Empirically, we have shown that generative oversampling works well for a range of text classification datasets using linear SVMs. Moreover,

if the best minority class training prior is unknown, generative oversampling has the added benefit of producing results which are robust to changes in minority class training prior. Generative oversampling is also simple to implement and can be used on a variety of different data types by selecting appropriate generative models. Therefore, it is a viable and flexible alternative whenever resampling methods are used.

We then analyzed the relationship between resampling and cost-sensitive learning. In particular, we examine the effect of random and generative oversampling versus cost-sensitive learning from a theoretical perspective using Bayesian classifiers. The theoretical analysis is supported by empirical results, where we briefly examined the effect of resampling on low-dimensional data and included a much more extensive treatment of resampling versus cost-sensitive learning on high-dimensional text datasets using multinomial naive Bayes and linear SVMs.

Results vary depending on the dataset and the classifier used. In particular, for low-dimensional datasets where a Gaussian distribution is appropriate to model the classes, there seems to be no advantage to using resampling. Theoretically, resampling results in the same expected sample mean but with greater variance. Empirically, there is no benefit to resampling over cost-sensitive learning when used with a Gaussian naive Bayes classifier. In practice, this means that resampling is unnecessary if Gaussian naive Bayes is used; a cost-sensitive classifier that performs just as well can easily be trained without the overhead of resampling. When applying multinomial naive Bayes to text datasets, resampling results in changed class priors as well as different estimates of the parameters of the multinomial distribution modeling the resampled class. In this case, any of the oversampling methods tested result empirically in better classification of the minority class. Finally, when classifying imbalanced text datasets using an SVM classifier, we see that using generative oversampling, which helps to expand the convex hull of the minority class,

can lead to consistently good performance. In particular, the best overall performance when classifying text datasets, regardless of classifier or method of resampling/incorporating costs, is generative oversampling coupled with SVMs.

Two of the most important results described in this chapter are as follows. First, while there is a theoretical equivalence between cost-sensitive learning and resampling under certain assumptions, we also show cases when these assumptions can be broken. For example, all of the experiments on high-dimensional text datasets (for both naive Bayes and SVMs) break these assumptions, leading to an observed empirical difference between cost-sensitive and resampling methods. We also show that there is no resampling method that is always best. We give analytical and empirical results supporting why different resampling methods interact differently with certain classifiers on certain types of data. Both of these results help explain why there are often differences between cost-sensitive learning and resampling methods in empirical studies such as [WMZ07]. Several areas of future work remain to explore other differences and further explain effects observed herein.

3.5 Connection to Active Learning

Finally, let us discuss the connection of these results to active learning. Our results show that there are cases where resampling and cost-sensitive learning will have essentially the same effect. In particular, this is true when the data can be modeled as Gaussian. One domain we will study extensively is hyperspectral data, where the data has been modeled well in empirical studies as a mixture of Gaussians. Thus, if one wants to handle cases of imbalance, the results in this chapter show that a more direct approach is to create a cost-sensitive active learner. This helps motivate the work in the next chapter, where, among other techniques, we introduce a method for performing cost-sensitive active learning.

Another idea previously proposed was to use resampling techniques to speed

up active learning. The basic idea is as follows. Given that there is an absolute scarcity of labeled points in active learning, a fast and inexpensive method of obtaining more points would be to use resampling. However, such an approach is flawed and ineffective if resampling does not change the learned model. The results in this chapter clarify when this will happen, and show that such an approach would be limited in scope. Thus, a cost-sensitive active learning method appears to be more useful.

Chapter 4

Uncertainty Sampling for Arbitrary Evaluation Metrics

4.1 Problem Description

In this chapter, we examine the problem of active learning. In particular, we introduce methods for modifying uncertainty sampling, a computationally effective method of active learning, to handle points with different misclassification costs and to handle problems which require different evaluation metrics. In order to do this, we also introduce a framework called "metric skew" for analyzing evaluation metrics, which not only allows for an uncertainty sampling approach that can handle different evaluation metrics, but also allows for several interesting conclusions to be drawn about evaluation metrics in machine learning.

4.2 Active Learning for Unequal Misclassification Costs

Active learning can be used to reduce the number of points required for a supervised learner to achieve a certain level of performance. As opposed to random sampling,

an active learning approach achieves this by letting the learning algorithm itself indicate which points should be labeled. Most studies on active learning assume that 0-1 loss is appropriate (i.e., all misclassifications are equally costly). However, in many problems, different classification errors will incur different costs. For example, in medical domains such as cancer detection, the misclassification costs are quite different for misdiagnosing a healthy versus unhealthy patient. An active learning approach that reduces 0-1 loss may not effectively lead to a classifier that reduces the total misclassification cost.

As discussed previously, in [Elk01], cost-sensitive learning is posed as a classification problem where misclassification costs are encoded in a cost matrix $\mathbf{C}$, where $C(i, j)$ is the cost for misclassifying a point from class j as a point from class i. The goal of cost-sensitive learning is to minimize the average misclassification cost of points in some unseen test set. More formally, let $\mathbf{x}$ represent an instance in the dataset and let $P(j|\mathbf{x})$ be the (estimated) posterior probability that $\mathbf{x}$ is in the jth class as predicted by some supervised learner. Then, the optimal classification as described in [Elk01] is to classify $\mathbf{x}$ as the class i that minimizes $\sum_j P(j|\mathbf{x})C(i, j)$, where $min_i \sum_j P(j|\mathbf{x})C(i, j)$ is called the class conditional risk.

This allows for a simple method of making classifiers that produce posterior probabilities (e.g., logistic regression, maximum likelihood) cost-sensitive. It is important, therefore, to have accurate posterior probability estimates in order to perform cost-sensitive learning. One active learning method which was designed to produce good posterior probability estimates is Bootstrap-LV [STP04]. We have found in preliminary experiments that Bootstrap-LV seems to suffer from problems similar to those described in section 4.2.2 and performs poorly (i.e., worse than random sampling) on text classification experiments with a multinomial maximum likelihood classifier (c4.5 was used in [STP04]). Thus, we do not consider Bootstrap-LV further.

In this section, we address the use of uncertainty sampling for reducing total misclassification cost. In contrast to loss-reduction methods, uncertainty sampling methods for active learning are computationally efficient. However, it is difficult to guide an uncertainty sampling approach to minimize arbitrary loss. We present a simple modification to uncertainty sampling based on self-training that allows for the efficient construction of cost-sensitive learners. We further show that several other naive methods of modifying uncertainty sampling to be cost-sensitive will not work well in general.

4.2.1 Uncertainty Sampling Versus Loss-reduction Methods

Pool-based active learning is an iterative process where, on each iteration of active learning, the active learner chooses n points from the currently unlabeled pool of training data $\mathcal{U}$, presents these n points to an oracle for labeling, and then adds these n recently labeled points to the labeled training data $\mathcal{L}$ (see [Set09] for a recent survey of active learning). Various approaches have been proposed for selecting points in $\mathcal{U}$, such as uncertainty sampling methods [LC94] and loss reduction methods [RM01].

To the best of our knowledge, most papers on handling misclassification costs for pool-based active learning pose the problem as an error-reduction or loss-reduction approach [RM01] [Mar05]. One exception is [STP07], which uses an uncertainty sampling approach for decision-centric learning, a problem related to cost-sensitive learning. However, [STP07] uses only the points selected for labeling during active learning to estimate posterior probabilities. We will show in the next section that this can result in poor estimates, particularly for generative classifiers.

An interesting approach for reducing various losses in a stream-based active learning setting has recently been introduced [BDL09]. However, a direct comparison between stream-based and pool-based approaches is outside the scope of this

dissertation. We should also note that, in contrast to more theoretically motivated papers such as [BDL09] and [DHM08] (among others) which focus on bounds, in this dissertation, we focus primarily on empirical results. This is not surprising since we modify uncertainty sampling, an active learning approach which has no strong theoretical guarantees, but has performed well empirically in many published studies.

As noted by many (e.g., see [Set09]), loss-reduction approaches for pool-based active learning are computationally costly since they require that a classifier be trained $|\mathcal{U}| * n_c$ times on every iteration of active learning, where n_c is the number of classes. While [RM01] offers advice on speeding up this process (e.g., subsampling, iterative learners), loss-reduction approaches can still be very computationally expensive.

The main advantage of using an uncertainty sampling approach over a loss reduction approach is computational efficiency. In this section, we present an approach which allows uncertainty sampling to handle different misclassification costs through the use of self-training [Yar95]. It is commonly known that active learning and semi-supervised learning are useful when unlabeled data is easier to obtain than labeled data, and in this section we show that there are additional benefits for cost-sensitive learning problems. Notably, in [DH08], a cluster-based active learning approach is proposed which also makes use of semi-supervised information; the authors also make many good comments on the problematic nature of "biased samples" obtained during active learning if one does not use semi-supervision.

4.2.2 Cost-sensitive Active Learning

Naive Methods of Cost-sensitive Uncertainty Sampling

In this section, we first list a number of simple approaches for incorporating misclassification costs into active learning. While at first glance, these may appear to be appropriate modifications to uncertainty sampling, all three of these methods are

problematic.

1. Naive Algorithm 1: Modify uncertainty scores to take costs into account; for example, instead of looking at margin between posterior probabilities, look at margin between class conditional risks

2. Naive Algorithm 2: Use a cost-sensitive classifier in the active learning process (i.e., pick points based on the uncertainty of a cost-sensitive learner)

3. Naive Algorithm 3: Pick points as in normal uncertainty sampling, but train a cost-sensitive classifier on $\mathcal{L}$; we will refer to this algorithm as "cost-sensitive uncertainty sampling", or "CS US"

All of these approaches need good, reliable posterior probability estimates. We will now show that posterior probabilities estimated from uncertainty sampling can be poor and unreliable, even for simple, easily separable problems.

Uncertainty Sampling and Posterior Probability Estimation

One can analytically show a case of active learning with a Bayesian classifier where the decision boundary is correct in expectation, but the expected values for the estimated parameters actually get worse as the active learning process progresses. We will show this for query-based learning[1], meaning that, in the following 1–d example, the active learner can ask for the label of any possible point $x \in \mathcal{R}$.

Assume a simple two-class problem where the class-conditional distributions of the positive and negative classes are Gaussian with means μ_+ and μ_- and with the same variance σ^2. Also assume that σ^2 and $P(y_+) = P(y_-) = 0.5$ are known. A maximum likelihood (ML) classifier is applied. Since σ^2 is known, the only parameters that need to be estimated are the means. Let $\hat{\mu}_{+,t}$ and $\hat{\mu}_{-,t}$ be the estimates

[1] as opposed to pool-based active learning, where only points that occur in $\mathcal{U}$, the unlabeled training data, can be labeled

of means μ_+ and μ_- after the t-th iteration of active learning. In appendix 6.2, we show that $E[\hat{\mu}_{\cdot,t}]$ actually moves away from the true means as t increases. One can also provide simple examples where this can also occur when σ^2 is estimated.

Continuing the above example, what happens to the estimated posterior probabilities as active learning progresses? One can show that the posterior probability of the positive class is of the form $P(y_+|x) = 1/(1 + exp(-\alpha x))$, where $\alpha \propto \frac{1}{\sigma^2}(\hat{\mu}_{-,t} - \hat{\mu}_{+,t})$. As t increases, $E[\hat{\mu}_{-,t} - \hat{\mu}_{+,t}]$ decreases. That is, as t increases, the "slope" of the sigmoid $P(y_+|x)$ decreases in expectation [2]. Thus, if one uses posterior probabilities as a measure of confidence, this means that, surprisingly, while the classification of points near the decision boundary remains unchanged, the confidence in these predictions actually decreases as t increases.

Let us consider a second, simple example 1–d dataset consisting of two, perfectly separable classes with equal priors distributed around the true decision boundary $x = 0$ such that all points greater than ϵ belong to the positive class and all points less than $-\epsilon$ belong to the negative class. The initial labeled dataset consists of $n_{+,0}$ positive examples and $n_{-,0}$ negative examples with means $\mu_+ > \epsilon$ and $\mu_- < -\epsilon$, respectively. Then, since we are running query-based active learning, on each iteration of active learning, one will draw points from either ϵ or $-\epsilon$, since these are the most uncertain points. For a given setting [3], Figure 4.1 plots the posterior probability $P(y_+|x)$ as a function of x obtained using Gaussian maximum likelihood for increasing number of samples drawn from ϵ and $-\epsilon$.

The posterior probability learned from Gaussian maximum likelihood actually becomes less certain as more points are sampled. One can show that the slope of the sigmoid $P(y_+|x)$ at $P(y_+|x) = 0.5$ is equal to $.25\frac{1}{\hat{\sigma}^2}(\hat{\mu}_{-,t} - \hat{\mu}_{+,t})$, where $\hat{\sigma}$ is the estimate of σ. For the values in the example in Figure 4.1, the distance between

[2] For the sake of brevity, we will informally refer to the slope of the sigmoid function at $P(y_+|x) = 0.5$ as the slope of the entire sigmoid function

[3] in this case, let $\mu_+ = 1$, $\mu_- = 1$, $n_{+,0} = 20$, $n_{-,0} = 20$, $\epsilon = 0.1$

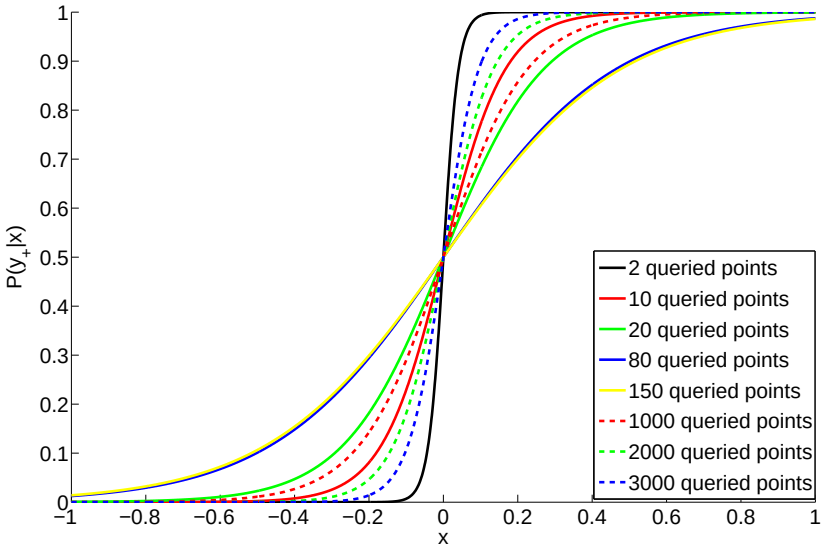


Figure 4.1: Example results for $\mu_+ = 1$, $\mu_- = -1$, $n_{+,0} = n_{-,0} = 20$, and $\epsilon = 0.1$.

the estimated means changes more rapidly than the estimated variance, causing the slope to initially decrease as more points are added. As $\hat{\sigma}^2$ eventually becomes small enough, the slope of the posterior probability slowly becomes sharper. However, the rate at which the sigmoid becomes sharper is quite slow, and it takes many samples to reach the same "confidence" learned with much fewer labeled points. Numerically, from Figure 4.1, one can see that, the slope of the posterior probability is initially quite "sharp" after just two queries. However, with just a few more queries, the slope of the posterior probability decreases significantly, and continues to decrease until around 150 points are added. Beyond 150 points, the variance begins to change more than the distance in means, causing the slope of the posterior probability to increase again. However, this process is quite slow, and even after querying 3000 points, the posterior probabilities near the decision boundary are still not as confident as probabilities obtained when the active learning process began. In short, the estimated posterior is unreliable, and the parameter estimates are poor.

Thus, if one uses posterior probability as a measure of confidence, one can show that, even for simple, completely separable cases, the confidence in predictions of points near the decision boundary can actually get worse as more points are labeled via uncertainty sampling for certain classifiers. Discriminative classifiers seem to be more immune to this problem, but it is difficult to make claims for such a broad class of classifiers.

For 0-1 loss, poor posteriors are not necessarily a problem, since one can have accurate classification even with poor posterior probability estimates. However, since posterior probability estimates are required to create cost-sensitive models for many classifiers, these poor posterior probabilities are particularly problematic when unequal misclassification costs are present.

Implications for Cost-sensitive Uncertainty Sampling

Since posterior probability estimates can be inaccurate due to the biased nature of uncertainty sampling, the naive methods of cost-sensitive uncertainty sampling listed above will fail on many datasets. Any variants of these algorithms (such as a combination of naive algorithms 1 and 2) which assume that the uncertainty sampling process results in a representative set of points will also perform poorly.

Empirically, naive methods 1 and 2 tend to perform very poorly across a wide range of datasets (naive method 1 is particularly poor). The third method, which picks points normally but trains a cost-sensitive classifier on the labeled points, works reasonably on many datasets (e.g., the results on hyperspectral data shown later in this section), but will perform very poorly on some datasets (e.g., the page-blocks dataset in figure 4.6 and the k1b dataset in figure 4.7). For the sake of brevity, we will not include results for naive methods 1 or 2. However, we will use the third method, which tended to outperform the other two methods in preliminary experiments, as a baseline which we will call "cost-sensitive uncertainty sampling"

or "CS US" for short.

4.2.3 Cost-sensitive Uncertainty Sampling with Self-training

Our proposed algorithm involves combining uncertainty sampling with self-training as follows:

1. **I**nput: Initial labeled set $\mathcal{L}$, unlabeled set $\mathcal{U}$
2. On each iteration of active learning, do the following:
   (a) Select n points from $\mathcal{U}$ based on uncertainty sampling; label and add to $\mathcal{L}$
   (b) Retrain classifier on $\mathcal{L}$
   (c) Classify all points in $\mathcal{U}$
   (d) Using known labels and points in $\mathcal{L}$ and predicted labels for $\mathcal{U}$, train a cost-sensitive classifier

We will refer to this algorithm as the "cost-sensitive uncertainty sampling with self-training" approach, or "CS USST" for short. In our experiments, we will also use a baseline known as "uncertainty sampling with self-training" (or "USST"), which is the same as "CS USST" except for the last step, where a cost-insensitive classifier is trained instead of a cost-sensitive classifier.

The motivation for this algorithm is that uncertainty sampling is useful for finding the decision boundary for 0-1 loss. However, as discussed in the previous section, the points picked by uncertainty sampling are unreliable for estimating model parameters and posterior probabilities. Thus, since uncertainty sampling is useful for finding 0-1 loss, the idea is to classify the unlabeled points in $\mathcal{U}$ and use these predicted class labels in a semi-supervised manner to find model parameters using all points in the training set.

An alternate motivation can be given by comparing the effect of uncertainty sampling with self-training against an uncertainty sampling approach with no self-training on the parameter estimates of a Gaussian maximum likelihood classifier. The regular uncertainty sampling approach will train on biased samples drawn from near the decision boundary. While the classes may be modeled well with Gaussian distributions, the biased samples near the decision boundary will not produce good estimates of mean and covariance. However, by self-training, one takes into account the entire set of points, meaning that the self-training approach will result in more accurate mean and covariance estimates since it includes more than just the biased sample of points near the decision boundary. Thus, in terms of the example given in figure 1, USST will result in more accurate posterior probability estimates than normal uncertainty sampling.

Note that there is no restriction in the algorithm that uncertainty sampling must be used. Thus, a query-by-committee or loss-reduction approach can be used instead. The incorporation of self-training should be useful for QBC methods [SOS92] since one can consider QBC methods to be uncertainty sampling methods where one is using an ensemble classifier instead of a single classifier. Preliminary experiments with loss-reduction show that combining loss-reduction methods with self-training reduces loss more quickly than just using loss-reduction. However, it should be emphasized that self-training is essential for producing a reliable cost-sensitive uncertainty sampling algorithm, but is merely beneficial for making loss-reduction algorithms efficient since it is already easy to incorporate misclassification costs into loss reduction methods.

In addition, the CS USST algorithm does not affect the selection of points. That is, the points selected by CS USST are no different than the points selected by a normal uncertainty sampling approach. Computationally, one only performs the final step of the CS USST algorithm when building the final classifier. If one

is simply labeling points via the active learning process, only one classifier needs to be retrained on each step of active learning.

Note that one could affect the selection of points by self-training on step 2b of the algorithm as well. However, initial experiments show that this computationally more expensive approach is not much better than the CS USST algorithm as presented, so we do not include results for such an algorithm.

4.2.4 Experiments

Experimental Setup

We present results on several domains of data, although the majority of results will be for hyperspectral datasets. These datasets are described more fully below.

In each of our experiments, we partition the data into training and test sets using five runs of ten-fold cross-validation and average the results. We use stratified sampling such that each fold has the same proportion of points from each class. Once the training and test sets have been created, 10% of the training data is randomly selected and labeled to form the initial labeled set $\mathcal{L}$, and the remaining unlabeled training data is placed in $\mathcal{U}$. On each iteration of active learning, 10 points are labeled.

In our experiments, we vary misclassification cost on the positive class between 1 and 25, and keep the misclassification cost on the negative class equal to 1. Because of lack of space, we only present results for $c_+ = 1$ (i.e., equal misclassification costs for all points) and $c_+ = 10$. For $c_+ > 1$, general trends on each dataset do not vary much, so results for other costs $c_+ > 1$ are similar to results for $c_+ = 10$.

In each experiment, we run a baseline algorithm called "cost-sensitive random sampling" or "CS RS" consisting of a cost-sensitive learner trained from randomly sampled points.

Datasets

The majority of our results are on the KSC and Botswana hyperspectral datasets described in Chapter 2. For the hyperspectral datasets, we run both LDA and logistic regression. We use uncertainty scores inversely proportional to the margin between the largest posterior probability and second largest posterior probability.

We also ran experiments on other domains, including various low-dimensional datasets and text classification datasets. The low-dimensional datasets consisted of data from the UCI data mining repository [AN07] and come from a number of domains. The main feature shared by these classifiers is that they consist of a relatively low-number of numerical features ("low" as compared to hyperspectral or text). We include results from the page-blocks and wisconsin-breast cancer datasets.

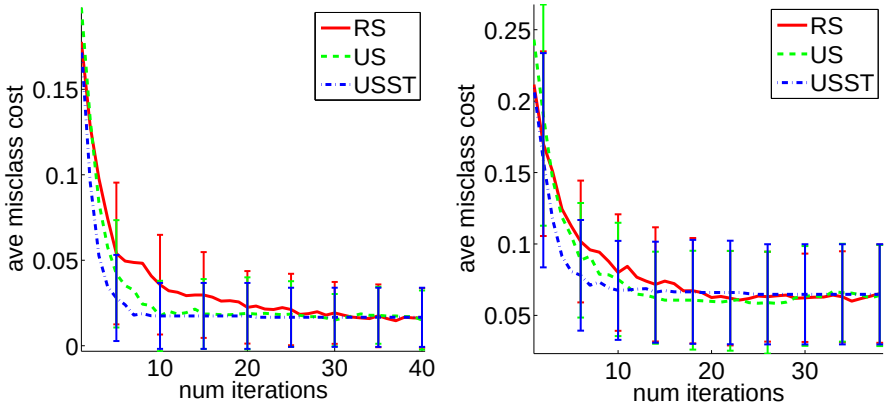
We also ran experiments on various text classification datasets. We include two sample results from the twenty newsgroups datasets and k1b data. For the k1b data, we concatenate several small classes to form a single positive and single negative class. For the twenty newsgroups dataset, we use two classes to create a two-class problem: comp.graphics and comp.windows.x.

For the low-dimensional data, we use an LDA classifier. For the text classification datasets, we use a maximum likelihood classifier that models each class with a multinomial distribution, a commonly used distribution for modeling text [MN98]. For both classifiers, uncertainty scores are inversely proportional to the margin between the largest posterior probability and second largest posterior probability.

Results

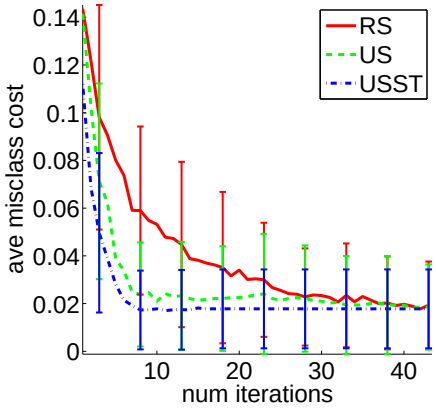
USST versus US

Let us first examine results for experiments where all misclassification costs are equal. The purpose of this section is to examine the effect of self-training on uncertainty sampling. Results for random sampling (RS), uncertainty sampling

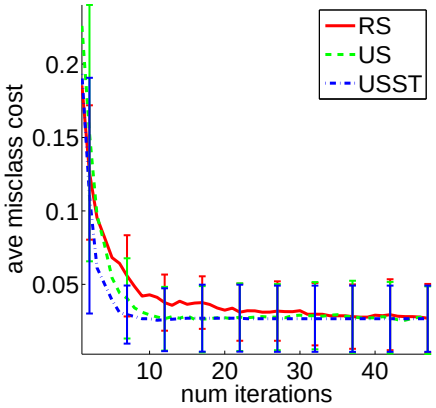


(a) KSC (class 3 vs 4) (*)

(b) KSC (class 4 vs 6)



(c) Botswana (class 5 vs 6) (*)

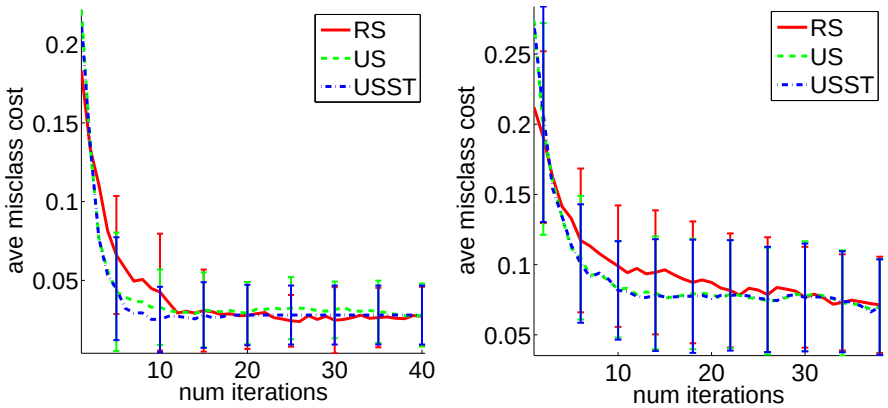


(d) Botswana (class 6 vs 9) (*)

Figure 4.2: Sample results for LDA on hyperspectral data; $c_+ = 1$, $c_- = 1$

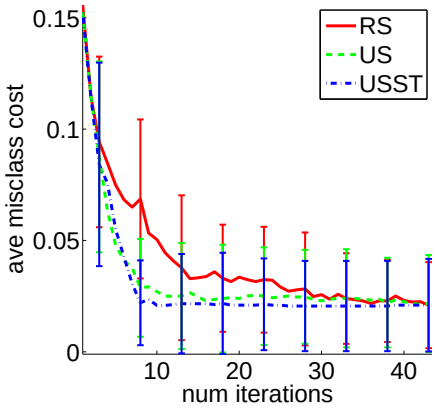
(US), and uncertainty sampling with self training (USST) are shown in figures 4.2 and 4.3 on hyperspectral data for LDA and logistic regression. Note that, in this case, RS, US, and USST are respectively equivalent to running CS RS, CS US, and CS USST with $c_+ = c_- = 1$.

Results are presented in the form of "banana curves", which are commonly used in presenting active learning results. In our figures, average misclassification cost on the test set is plotted against the number of iterations of active learning that have been performed. An alternate metric is to calculate the relative reduction in loss compared to random sampling. We calculate this by looking at the area between

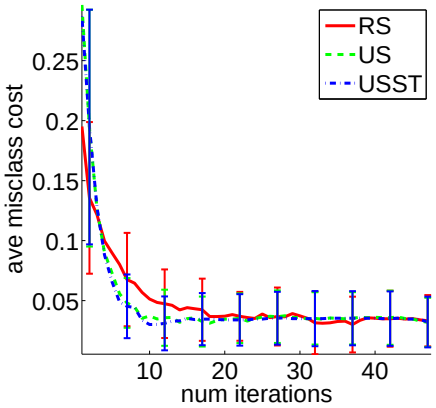


(a) KSC (class 3 vs 4) (*)

(b) KSC (class 4 vs 6)



(c) Botswana (class 5 vs 6) (*)

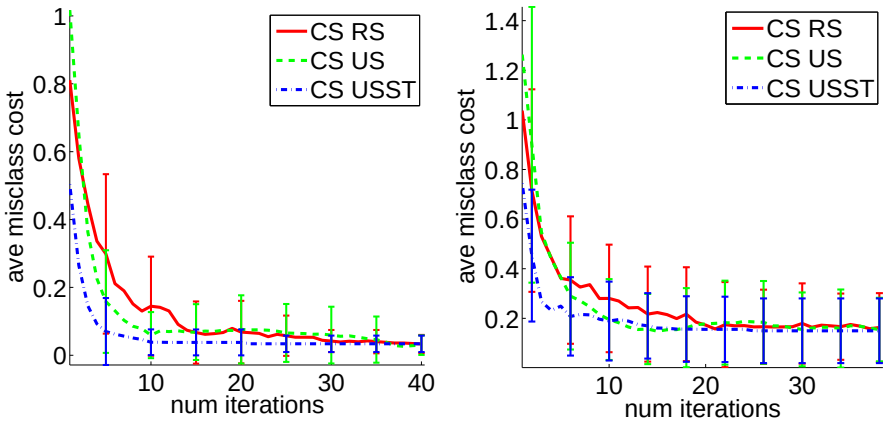


(d) Botswana (class 6 vs 9)

Figure 4.3: Sample results for logistic regression on hyperspectral data; $c_+ = 1$, $c_- = 1$

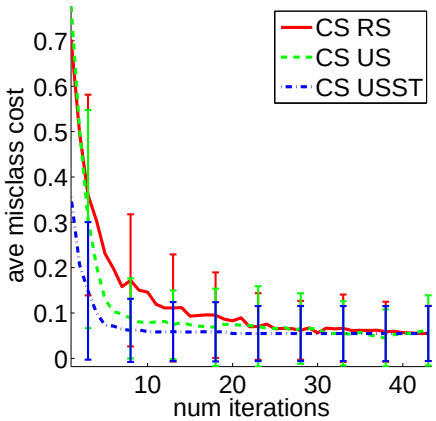
a particular active learning curve and the curve obtained for random sampling. In figures 2 through 7, we will use an asterisk to denote results where the difference in these areas for CS US and CS USST are statistically significant as determined by a t-test with p-value of 0.05.

As one can see in the figures, USST consistently outperforms standard uncertainty sampling and random sampling. In particular, USST converges to the misclassification cost obtained after all of $\mathcal{U}$ has been labeled much faster than either standard uncertainty sampling or random sampling. The difference between USST and US is much larger for LDA, which inherently assumes that the classes are

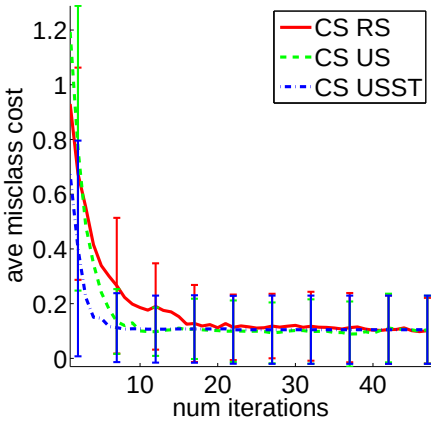


(a) KSC (class 3 vs 4) (*)

(b) KSC (class 4 vs 6) (*)



(c) Botswana (class 5 vs 6) (*)



(d) Botswana (class 6 vs 9) (*)

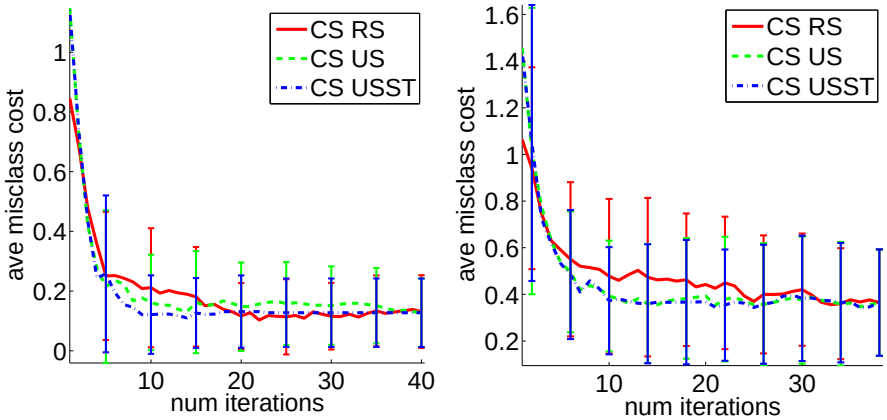
Figure 4.4: Sample results for LDA on hyperspectral data; $c_+ = 10$, $c_- = 1$

Gaussian, than for logistic regression. This indicates that self-training is more useful for classifiers where some assumption about the distribution of the entire dataset is made.

Regardless of the classifier, USST converges faster or at the same rate as US in these experiments. Thus, even when all misclassification costs are equal, USST is preferable to uncertainty sampling.

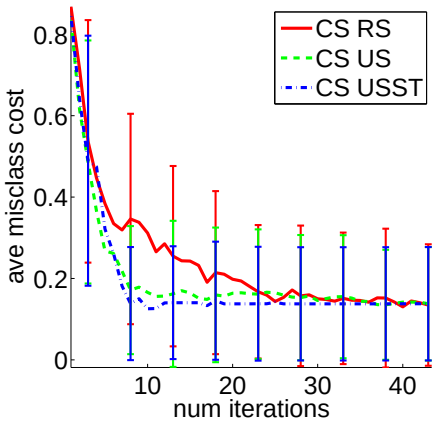
CS USST versus CS US

In this section, we will examine the performance of CS USST. We will com-

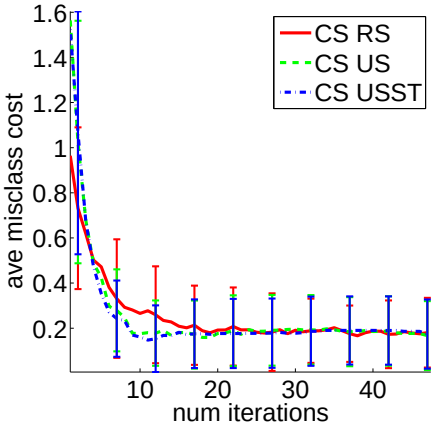


(a) KSC (class 3 vs 4) (*)

(b) KSC (class 4 vs 6)



(c) Botswana (class 5 vs 6)



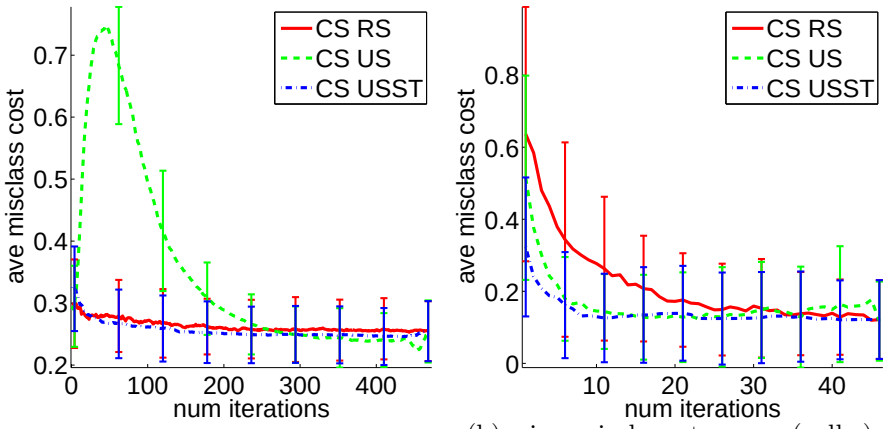
(d) Botswana (class 6 vs 9)

Figure 4.5: Sample results for logistic regression on hyperspectral data; $c_+ = 10$, $c_- = 1$

pare CS USST against cost-sensitive random sampling (CS RS) and cost-sensitive uncertainty sampling (CS US).

Results for hyperspectral data are shown in figure 4.4 for LDA. As mentioned, results tend to be similar regardless of the ratio between c_+ and c_-. From the figures, it is evident that CS USST consistently outperforms the other methods, and all differences between CS USST and CS US are statistically significant.

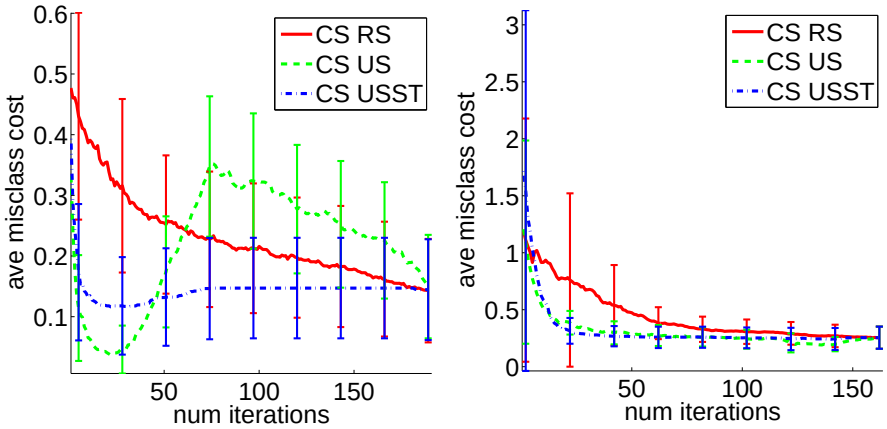
Results for the hyperspectral datasets are shown in figure 4.5 for a logistic regression classifier. Since the posterior probabilities estimated using logistic



(a) page blocks (*)

(b) wisconsin breast cancer (wdbc) (*)

Figure 4.6: Sample results for LDA on low-dimensional data; $c_+ = 10$, $c_- = 1$



(a) k1b (*)

(b) 20 newsgroups (comp.graphics vs comp.windows.x)

Figure 4.7: Sample results for multinomial maximum likelihood on text data; $c_+ = 10$, $c_- = 1$

regression based on the points selected via uncertainty sampling tend to be fairly accurate, there is not much difference between results obtained for CS USST and CS US. However, CS USST is still consistently better than or equal in performance compared to CS US.

As in the results for $c_+ = 1$, the difference between CS USST and CS US are

greater for LDA than for logistic regression. Again, this indicates that self-training is more useful for cases where some assumption about the entire distribution of the data is made.

We have also performed experiments using LDA on various low-dimensional datasets. Sample results are shown in figure 4.6. Note that, on the page-blocks dataset, CS US actually does very poorly, fluctuating first below then far above the random baseline. In contrast, the performance of CS USST is much more reliable.

Sample results for a multinomial maximum likelihood classifier on text datasets are shown in figure 4.7. Like the low-dimensional datasets, there is a case where CS US fluctuates wildly, but CS USST does not (the k1b dataset). In addition, these results indicate that CS USST can work well for high-dimensional datasets and cases where the data is clearly not Gaussian.

Thus, while CS US can work well on some problems (e.g., the hyperspectral domain), it is not reliable, and will perform erratically on some problems (e.g., the k1b dataset and page-blocks dataset) where CS USST can still perform well.

4.3 Metric Skew: A Framework for Analyzing Evaluation Metrics

In this section, we will describe a framework for analyzing evaluation metrics. While this may seem unrelated to active learning, we will show in the next section (section 4.4) that one way of looking at this analysis framework can be combined with the cost-sensitive uncertainty sampling method described above in order to create an uncertainty sampling method that can optimize any evaluation metric that can be described as a function of terms in a confusion matrix.

Many evaluation metrics used to analyze the results in classification problems do not weight classifications from all classes equally. The most obvious example of

this are evaluation metrics used in cost-sensitive learning which specifically take disparate misclassification costs into account. However, several metrics that do not use misclassification costs are also not symmetric with respect to the effect of correctly classifying points from different classes. Table 4.1 illustrates a simple example where this is true for a variety of common evaluation metrics. We look at the values of the metrics on two related two-class problems. In both cases, the number of points in the positive and negative classes are both equal to 100. In case 1, the number of true positives is 80 while the number of true negatives is 50; in case 2, the opposite is true: the number of true positives is 50 while the number of true negatives is 80. Note that only accuracy is equal in both cases, while for all other metrics, there is clearly a different effect due to misclassifications in the positive versus the negative class.

Table 4.1: A simple example of asymmetry

Metric	Case 1	Case 2
accuracy	0.65	0.65
precision	0.62	0.71
recall	0.80	0.50
f1-measure	0.70	0.59

Several authors have noted that many evaluation metrics do not weight misclassification costs equally (e.g., [SJS06]). In this section, we introduce a framework for analyzing evaluation metrics that quantifies the impact of correctly classifying samples from each class. We then apply the framework to a number of common metrics. We show that our framework provides a new viewpoint on several known and previously unknown properties about metrics.

4.3.1 Evaluation Metrics

A number of popular evaluation metrics have been introduced in the past for analyzing the predictions of classification algorithms. Below, we describe several common metrics analyzed in this dissertation. In particular, we will show that many common metrics on two-class problems can be expressed as a function $\boldsymbol{m}$ of number of true positives n_{tp}, number of true negatives n_{tn}, and constants.

For the two class case, if an evaluation metric can be defined as a function of entries in a confusion matrix for a two class problem, then one can define the evaluation metric as a function of two variables and various constants. We choose to use n_{tp} and n_{tn} as the two variables (typically, n_+ and n_- will be treated as constants, although these may also need to be estimated if one does not know the true prior distributions). We can therefore express evaluation metrics as a function $\boldsymbol{m}(n_{tp}, n_{tn})$.

For cases with more than two classes, our approach can handle any evaluation metric that is a function of terms in a confusion matrix as long as the classifier used can handle a cost matrix $\mathbf{C}$ of the same size. This will become clearer in section 4.3.2 when our approach is defined. First, however, let us define each metric as a function $\boldsymbol{m}$.

Total Cost and Variants

As discussed throughout this dissertation, if misclassification costs are known, a useful evaluation metric is the total misclassification cost.

For a two-class problem, let the cost of making a false positive be denoted as c_{fp} and similarly for true positives (c_{tp}), true negatives (c_{tn}), and false negatives

(c_{fn}). Then the total cost is defined as [4]:

$$\begin{aligned} \boldsymbol{m}_{totalcost}(n_{tp}, n_{tn}) &= c_{tp}n_{tp} + c_{tn}n_{tn} \\ &\quad + c_{fn}n_{fn} + c_{fp}n_{fp} \end{aligned} \tag{4.1}$$

In practice, c_{tp} and c_{tn} are often set to zero so as not to penalize the classifier for correct decisions. c_{fp} and c_{fn} are also normalized by c_{fp} such that costs c'_{fp} and c'_{fn} are used, where $c'_{fp} = 1$ and $c'_{fn} = \frac{c_{fn}}{c_{fp}}$. In this case, equation 4.1 can be rewritten as:

$$\begin{aligned} \boldsymbol{m}'_{totalcost}(n_{tp}, n_{tn}) &= c'_{fn}n_{fn} + n_{fp} \\ &= c'_{fn}(n^+ - n_{tp}) + n^- - n_{tn} \end{aligned} \tag{4.2}$$

Note that if equation 4.2 is divided by n, then we get the expected cost. In addition, if $c'_{fn} = 1$, then the expected cost becomes 1 minus the average accuracy of the classifier.

ROC and AUC

A ROC (receiver operating characteristic) curve can be thought of as a graphical metric used to evaluate classifiers. The horizontal axis is used to plot the false positive rate ($\frac{n_{fp}}{n_-}$) assigned by a particular classifier setting and the vertical axis is used to plot the true positive rate ($\frac{n_{tp}}{n_+}$) at the same classifier setting.

A convenient way of comparing ROC curves is the area under the curve (AUC), an aptly named numerical metric obtained by finding the area under the ROC curve. In practice, since only certain points on the ROC curve are found empirically, the AUC can be conveniently calculated using the trapezoid rule.

[4] Note that defining cost in terms of n_{tp} and n_{tn} instead of n_{fn} and n_{fp} is somewhat unnatural; we do this such that all metrics are defined as functions of the same variables

We define AUC as follows in order to analyze AUC for a single value of n_{tp} and n_{tn}. Note that the current values of n_{tp} and n_{tn} only affect one part of the ROC curve. Let (n_{tp1}, n_{tn1}) be the point on the ROC curve immediately to the left of (n_{tp}, n_{tn}) and let (n_{tp2}, n_{tn2}) be the point immediately to the right of (n_{tp}, n_{tn}) on the ROC curve. Then, using the trapezoid rule, the contribution to total AUC based on the current point (n_{tp}, n_{tn}) and its two immediate neighbors is given by:

$$\begin{aligned} \boldsymbol{m}_{pAUC}(n_{tp}, n_{tn}) &= \\ & \frac{1}{2}\left(\frac{n_{tn1}}{n^-} - \frac{n_{tn}}{n^-}\right)\left(\frac{n_{tp}}{n^+} + \frac{n_{tp1}}{n^+}\right) \\ &+ \frac{1}{2}\left(\frac{n_{tn}}{n^-} - \frac{n_{tn2}}{n^-}\right)\left(\frac{n_{tp}}{n^+} + \frac{n_{tp2}}{n^+}\right) \end{aligned} \tag{4.3}$$

where the subscript "pAUC" is meant to highlight the fact that this only calculates part of the total AUC.

Note that the ROC curve always includes the two trivial classifiers that always predict the negative class or always predict the positive class. Thus, a trivial three-point ROC curve can be created for any single value of (n_{tn}, n_{tp}) by letting $(n_{tp1}, n_{tn1}) = (0, n^-)$ and $(n_{tp2}, n_{tn2}) = (n^+, 0)$. In addition, for the trivial three-point ROC curve, $\boldsymbol{m}_{pAUC}$ is equal to the AUC of the entire ROC curve.

Precision, Recall, and Variants

For a two-class problem, precision and recall are defined as follows:

$$\begin{aligned} \boldsymbol{m}_{precision}(n_{tp}, n_{tn}) &= \frac{n_{tp}}{n_{tp} + n_{fp}} \\ &= \frac{n_{tp}}{n_- + n_{tp} - n_{tn}} \end{aligned} \tag{4.4}$$

$$\boldsymbol{m}_{recall}(n_{tp}, n_{tn}) = \frac{n_{tp}}{n_{tp} + n_{fn}} = \frac{n_{tp}}{n_+} \tag{4.5}$$

Since it is convenient to look at a single metric instead of two separate metrics, precision and recall have been combined in a number of ways. One popular method is the f1-measure, the harmonic mean of precision and recall, which is equal to:

$$\boldsymbol{m}_{f1-measure}(n_{tp}, n_{tn}) = \frac{2 * n_{tp}}{n + n_{tp} - n_{tn}} \tag{4.6}$$

Another method of combining precision and recall is to take the geometric mean.

To extend precision and recall for multi-class problems, either the microaverage or macroaverage can be used. The microaveraged recall is defined as:

$$\boldsymbol{m}_{mic.rec.} = \frac{\sum_{j=1}^{k} n_{tp}^{j}}{\sum_{j=1}^{k} n^{j}} \tag{4.7}$$

while macroaveraged recall is defined as:

$$\boldsymbol{m}_{mac.rec.} = \frac{1}{k} \sum_{j=1}^{k} \frac{n_{tp}^{j}}{n^{j}} \tag{4.8}$$

Similarly, microaveraged precision is defined as:

$$\boldsymbol{m}_{mic.prec.} = \frac{\sum_{j=1}^{k} n_{tp}^{j}}{\sum_{j=1}^{k} n_{tp}^{j} + \sum_{j=1}^{k} n_{fp}^{j}} \tag{4.9}$$

while the macroaveraged precision is defined as:

$$\boldsymbol{m}_{mac.prec.} = \frac{1}{k} \sum_{j=1}^{k} \frac{n_{tp}^{j}}{n_{tp}^{j} + n_{fp}^{j}} \tag{4.10}$$

4.3.2 Metric Skew

In this section, we present our approach for determining the effect of classifying an additional point from a specific class on the value of an evaluation metric. The method involves finding the derivative of a metric $\mathbf{m}$ with respect to changes in the number of correctly classified instances from that class.

Let us begin by looking at the simplest possible case: a two-class problem. In order to determine how much the metric will change if we add an additional h true positives or h true negatives, we can define the following delta functions:

$$\boldsymbol{\delta}_+(n_{tp}, n_{tn}, h) = \frac{\boldsymbol{m}(n_{tp} + h, n_{tn}) - \boldsymbol{m}(n_{tp}, n_{tn})}{h} \tag{4.11}$$

and

$$\boldsymbol{\delta}_-(n_{tp}, n_{tn}, h) = \frac{\boldsymbol{m}(n_{tp}, n_{tn} + h) - \boldsymbol{m}(n_{tp}, n_{tn})}{h} \tag{4.12}$$

where $\boldsymbol{m}(n_{tp}, n_{tn})$ is the value of some evaluation metric $\boldsymbol{m}$ given n_{tp} and n_{tn}. Thus, a metric will change by $\boldsymbol{\delta}_+(n_{tp}, n_{tn}, 1)$ if the number of true positives is increased by 1 and will change by $\boldsymbol{\delta}_-(n_{tp}, n_{tn}, 1)$ if the number of true negatives is increased by 1.

If we take the limit of $\boldsymbol{\delta}_+(n_{tp}, n_{tn}, h)$ as h approaches 0, we get the derivative of the metric with respect to n_{tp}. We denote this derivative as $\frac{\partial \boldsymbol{m}}{\partial n_{tp}}$. Similarly, $\frac{\partial \boldsymbol{m}}{\partial n_{tn}}$ is equal to the limit of $\boldsymbol{\delta}_-(n_{tp}, n_{tn}, h)$ as h approaches 0. Note that this method can be easily extended to the general multi-class case by computing the derivatives of the metric with respect to the number of points correctly classified in each class.

Let us define the skew of a metric towards the jth class as $\frac{\partial \mathbf{m}}{\partial n_{tp}^j}$. For the special two-class case, we call the derivatives $\frac{\partial \mathbf{m}}{\partial n_{tp}}$ and $\frac{\partial \mathbf{m}}{\partial n_{tn}}$ the metric skew towards the positive and negative classes, respectively. We use the term "skew" for a number of reasons. The first is to avoid confusion with other potential names, such as cost or bias, which are already commonly used for other purposes. The second is that the

term "skew" has the connotation of a slant or bias towards a particular direction. A third reason is its connection with a quantity known as the effective skew ratio [Fla03] which we will describe in more detail below.

First, however, let us define the skew ratio of a metric for two-class problems. Let the skew ratio be defined as the skew towards the positive class divided by the skew towards the negative class (i.e., skew ratio= $\frac{\partial \mathbf{m}/\partial \mathbf{n_{tp}}}{\partial \mathbf{m}/\partial \mathbf{n_{tn}}}$). If the skew ratio is greater than one, then increasing the number of true positives will increase the value of the metric more than increasing the number of true negatives; that is, the metric is skewed towards the positive class. The skew ratio is a convenient quantity when looking at metric skews in two-class problems. However, while metric skews can be defined for multi-class problems, the skew ratio cannot. For multi-class problems, we must compare individual metric skews against each other (we will show an example of this for micro/macro averaged precision/recall).

In [Fla03], a quantity known as the effective skew ratio was defined. The effective skew ratio is the slope of a metric's isometric lines in ROC space and is related to the skew ratio by the following theorem:

Theorem 4.3.1. *For a two-class problem, the effective skew ratio of a metric is equal to $\frac{n^-}{n^+}$ times the inverse of the skew ratio.*

Proof. Denote a metric in ROC space as a function $\boldsymbol{m}_{ROC}(n_{tpr}, n_{fpr})$ where the true positive rate is defined as $n_{tpr} = \frac{n_{tp}}{n^+}$ and the false positive rate is defined as $n_{fpr} = \frac{n_{fp}}{n^-} = \frac{n^- - n_{tn}}{n^-}$. $\boldsymbol{m}_{ROC}(n_{tpr}, n_{fpr}) = \boldsymbol{m}(n^+ n_{tpr}, n^- - n^- n_{fpr})$ where $\boldsymbol{m}$ is a metric defined as some function of n_{tp} and n_{tn} as done previously in this section.

An isometric line of a metric in ROC space is defined by letting $\boldsymbol{m}_{ROC}(n_{tpr}, n_{fpr}) = m'$ where m' is some constant. The effective skew ratio is the slope of this line in ROC space and is equal to $\frac{dn_{tpr}}{dn_{fpr}}$.

In calculus, implicit differentiation states the following: If a differentiable function $\mathbf{f}(x, y) = 0$, where y is defined implicitly as a differentiable function of x,

and $\frac{\partial \mathbf{f}}{\partial y} \neq 0$, then $\frac{dy}{dx} = -\frac{\partial \mathbf{f}/\partial x}{\partial \mathbf{f}/\partial y}$.

Let $\mathbf{f}(n_{tpr}, n_{fpr}) = \boldsymbol{m}_{ROC}(n_{tpr}, n_{fpr}) - m' = 0$. Then, because of implicit differentiation, the effective skew ratio $\frac{dn_{tpr}}{dn_{fpr}}$ is equal to $-\frac{\partial \mathbf{f}/\partial n_{fpr}}{\partial \mathbf{f}/\partial n_{tpr}}$.

Note that: $\frac{\partial \mathbf{f}}{\partial n_{tpr}} = \frac{\partial \boldsymbol{m}_{ROC}}{\partial n_{tpr}} = \frac{\partial \boldsymbol{m}}{\partial n_{tp}} \frac{dn_{tp}}{dn_{tpr}} = n^+ \frac{\partial \boldsymbol{m}}{\partial n_{tp}}$ and $\frac{\partial \mathbf{f}}{\partial n_{fpr}} = \frac{\partial \boldsymbol{m}_{ROC}}{\partial n_{fpr}} = \frac{\partial \boldsymbol{m}}{\partial n_{tn}} \frac{dn_{tn}}{dn_{fpr}} = -n^- \frac{\partial \boldsymbol{m}}{\partial n_{tn}}$

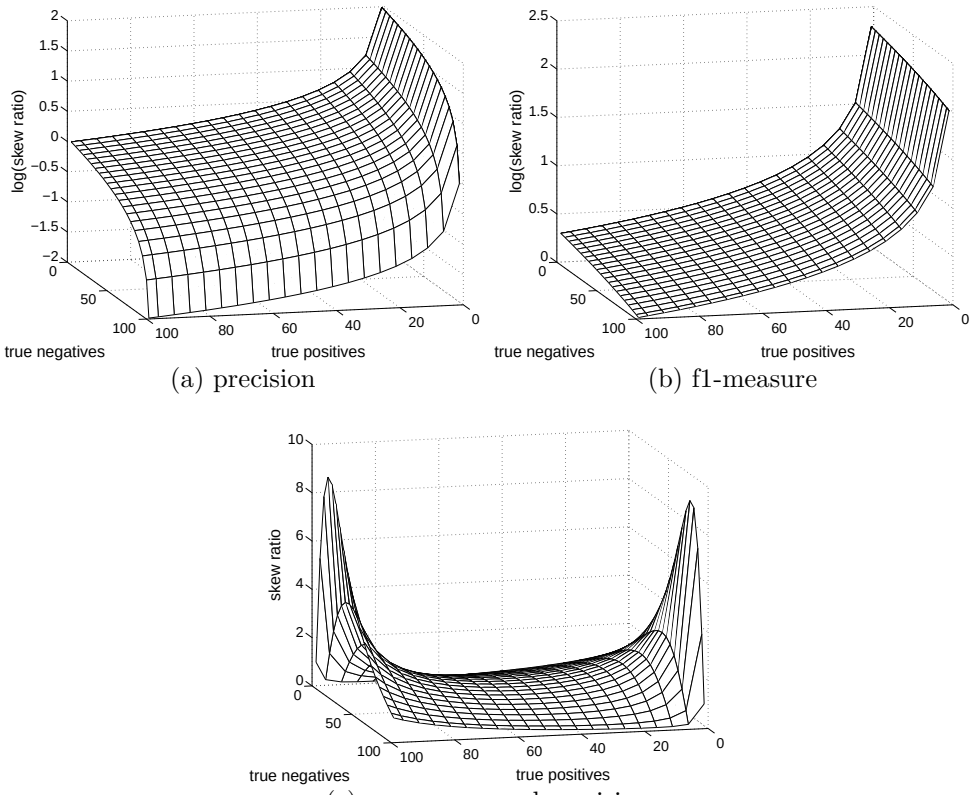
Thus, effective skew ratio $= \frac{dtpr}{dfpr} = -\frac{\partial \mathbf{f}/\partial fpr}{\partial \mathbf{f}/\partial tpr} = \frac{n^-}{n^+} \frac{\partial \boldsymbol{m}/\partial n_{tn}}{\partial \boldsymbol{m}/\partial n_{tp}} = \frac{n^-}{n^+} \frac{1}{\text{skew ratio}}$.

□

Both the effective skew ratio and metric skew can be used for analyzing and characterizing metrics. However, using metric skew has several advantages. First, since determining the effective skew ratio depends on isometrics in ROC space, it is more difficult to extend beyond two-class problems (we are unaware of any such extensions). However, metric skews can be calculated for each class in a multi-class problem as long as the metric is defined in terms of constants and n^j_{tp} for all classes j. Furthermore, the method of calculating metric skew is more straightforward than the example method of finding effective skew ratios based on isometrics in ROC space as described in [Fla03]. Finally, for two-class problems specifically, using the ratio of the two metric skews for each class (the skew ratio) rather than effective skew ratio is better suited for determining whether a metric is biased towards the positive or negative class. For example, if the skew ratio is exactly equal to 1, then the metric is not skewed towards either class. However, a skew ratio of 1 means that the metric has an effective skew ratio of $\frac{n^-}{n^+}$. Here, the effects of classifying the positive and negative class is less straightforward to interpret.

4.3.3 Analysis of Common Metrics

Let us now use metric skew to determine the skew ratios of several common metrics in two-class problems. We will also use metric skew to examine some example metrics for multi-class problems. We will see that our approach not only confirms



(a) precision (b) f1-measure

(c) macroaveraged precision

Figure 4.8: Example graphs of non-constant skew ratios when $n^+ = 100$, $n^- = 100$

and is consistent with known observations on evaluation metrics, but also provides several new properties.

A summary of skew ratios for two-class problems are in table 4.2. Figure 4.8 plots some of the non-constant skew ratios from table 4.2. In all plots in figure 4.8, $n^+ = n^- = 100$ while n_{tp} and n_{tn} are varied. Note that the plots show the log of the skew ratio for precision and f1-measure, but not macroaveraged precision.

Total Cost and Variants

First, let us show that our method of finding metric skew is consistent with cost-sensitive metrics. Using equation 4.2 as the evaluation metric, $\frac{\partial \mathbf{m}}{\partial n_{tp}} = -c'_{fn}$ and $\frac{\partial \mathbf{m}}{\partial n_{tn}} = -1$. Thus, the skew ratio equals c'_{fn}, meaning that misclassifying the positive class is c'_{fn} times more costly than misclassifying the negative class, a result that is exactly consistent with the definition of costs in cost-sensitive learning. Similarly, it is easy to show that the skew ratio of average accuracy is equal to 1, which is consistent with the fact that true positives and true negatives contribute equally to average accuracy.

AUC

For AUC, we get a particularly interesting skew ratio. Using equation 4.3 (which only calculates the contribution of the current values of n_{tp} and n_{tn} to the total AUC based on its immediate neighbors), we obtain a skew ratio of $\frac{n_{tn1}-n_{tn2}}{n_{tp2}-n_{tp1}}$, meaning that the relative importance of correctly classifying the positive versus the negative class depends strictly on the values in the ROC curve to the immediate left and right of the current point. As discussed, a single value of n_{tp} and n_{tn} returned by a classifier can be turned into a three-point ROC curve by including the trivial classifier that always guesses the negative class and the classifier that always guesses the positive class. In this special case, the skew ratio is equal to $\frac{n^-}{n^+}$, meaning that correct classifications from each class are weighted proportionally to the inverse of its prior.

Precision, Recall, and Variants

For recall, the skew for the positive class is $1/n^+$ and the skew for the negative class is 0. Since recall depends only on performance on the positive class, it is always more important to do well on the positive class, which is consistent with the values for the metric skews. The skew ratio of precision is $\frac{n^- - n_{tn}}{n_{tp}}$; the log of this skew

ratio is plotted in Figure 4.8(a).

For f1-measure, the skew ratio is equal to $\frac{n-n_{tn}}{n_{tp}}$. Since $n \geq n_{tn} + n_{tp}$, the skew ratio is always greater than or equal to 1, with equality only if there are no false positives and no false negatives. At this point, of course, there is no way to increase f1-measure, meaning that increasing the number of true positives always has a greater effect than increasing the number of true negatives. In Figure 4.8(b), the log of the skew ratio of f1-measure is plotted. Note that the skew ratio is largest when there are very few true positives and smallest when n_{tp} and n_{tn} are close to n^+ and n^-.

The skew ratio of the geometric mean of precision and recall is equal to $\frac{2n^- + n_{tp} - 2n_{tn}}{n_{tp}}$. Since the maximum value of n_{tn} is n^-, the skew ratio is also always greater than or equal to 1.

Analysis of Metrics for Multi-class Problems

Now let us use the microaveraged and macroaveraged precision and recall as examples of how our approach can be applied to the more general multi-class case. In order for a more direct comparison with our previous analysis on two-class metrics, the skew ratios of these metrics are also described below and included in table 4.2.

The skew of microaveraged recall for the jth class is $\frac{1}{n}$. Since this is a constant, the skew of microaveraged recall for all classes is equal. Thus, all classifications contribute equally to microaveraged recall, and, for the two-class case, the skew ratio is 1. The skew of macroaveraged recall for the jth class is $\frac{1}{n^j}$. That is, classifications are weighted with weights equal to the inverse of the number of points in that class. In the two-class case, this means the skew ratio is $\frac{n^-}{n^+}$. This is consistent with previous work.

The skew of the microaveraged precision is also a constant, meaning that all classifications contribute equally to microaveraged precision. However, the skew

Table 4.2: A summary of evaluation metrics and their skew ratios

Metric	Skew ratio
accuracy	1
total cost	c'_{fn}
partial AUC	$\frac{n_{tn1}-n_{tn2}}{n_{tp2}-n_{tp1}}$
recall	see discussion
precision	$\frac{n^- - n_{tn}}{n_{tp}}$
f1-measure	$\frac{n - n_{tn}}{n_{tp}}$
geo. mean of prec. and rec.	$\frac{2n^- + n_{tp} - 2n_{tn}}{n_{tp}}$
microaverage recall	1
macroaverage recall	$\frac{n^-}{n^+}$
microaverage precision	1
macroaverage precision	see discussion

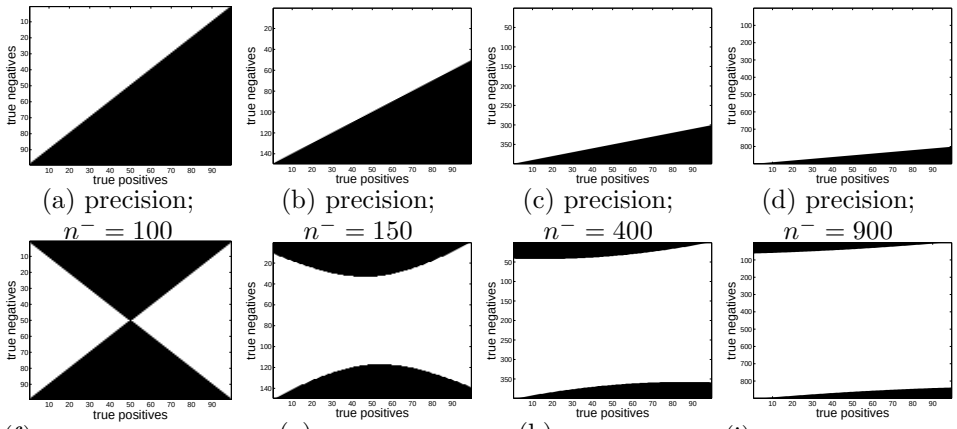
of macroaveraged precision is not the same as the skew of macroaveraged recall and is much less straightforward. For the simplest two-class case, the skew ratio of macroaverage precision is equal to:

$$\frac{n^-(n^+ + n_{tn} - n_{tp})^2 + n_{tn}(n)[n^- - n^+ + 2(n_{tp} - n_{tn})]}{n^+(n^- + n_{tp} - n_{tn})^2 + n_{tp}(n)[n^+ - n^- + 2(n_{tn} - n_{tp})]}$$

which is in general not equal to $\frac{n^-}{n^+}$. This skew ratio is plotted in Figure 4.8(c) for the case where $n^+ = n^- = 100$. As evident from the figure, the skew ratio of macroaverage precision is a saddle function. Thus, macroaveraged precision does not weight classifications inversely proportional to the number of points from that class in the test set. Instead, macroaveraged precision weights classifications very differently depending on the current number of correct classifications.

4.3.4 Discussion

Our work shows that care must be taken when misclassification costs are either equal or unknown. If misclassification costs are equal, then care must be exercised when using and interpreting results from a metric with a skew ratio that is not 1.



(a) precision; $n^- = 100$ (b) precision; $n^- = 150$ (c) precision; $n^- = 400$ (d) precision; $n^- = 900$

(f) macroave. prec.; $n^- = 100$ (g) macroave. prec.; $n^- = 150$ (h) macroave. prec.; $n^- = 400$ (i) macroave. prec.; $n^- = 900$

Figure 4.9: Graphs of indicator function for whether skew ratio>1 for $n^+ = 100$ and for different values n^-. On the graph, white indicates skew ratio>1 and black indicates skew ratio$<=1$. Note that even for moderate amounts of imbalance, both example metrics are skewed towards the positive class for most possible values of n_{tp} and n_{tn}.

For example, in a two-class problem, precision, recall, and f-measure are often used in cases where both classes are equally important; however, this might lead to incorrect conclusions. In cases where classes are equally important, the *microaveraged* precision, recall, and f-measure should be used instead.

In cases where misclassification costs are unequal and unknown, the use of certain evaluation metrics with certain values of skew results in an implicit misclassification cost for each class that is equal to the skew for that class. An example where misclassification costs are unequal and unknown is the imbalanced dataset problem discussed in the previous chapter.

As discussed previously, in the imbalanced dataset problem, the priors of each class are highly unequal. For example, in a two-class case, the probability of a datapoint from the positive class is much smaller than the probability of a datapoint from the negative class. Without accounting for class imbalance, many classifiers often learn to assign all points to the negative class. The argument is that this type

of classifier is well-nigh useless, and, in example domains such as cancer detection, various types of fraud detection, and intrusion detection, this argument holds true.

However, if misclassification costs are equal, a classifier that learns to always predict the class with the highest prior may, in fact, be a very good learned classifier. For example, if misclassification costs are equal and the prior probability of the negative class is, say, 99%, then a classifier that always assigns points to the negative class is just as valid as a classifier which is completely correct on the positive class but misclassifies 1% of the data points (i.e., 1/99th of the negative class). While the second classifier seems more palatable, the total number of misclassifications is the same in both cases [5]. In practice, it may also be difficult to increase the number of true positives without introducing a large number of false positives as well, meaning that a classifier which always guesses the negative class may in fact lead to the lowest number of total misclassifications. Thus, a classifier learned from an imbalanced dataset which always predicts a single class is a problem only when misclassifying the minority class has some cost that is higher and unequal to the cost of misclassifying the majority class.

Thus, the imbalanced dataset problem involves both unequal priors as well as a (possibly unknown) higher misclassification cost for the minority class. Many past papers on imbalanced datasets have stated that an evaluation metric such as total accuracy or total number of misclassifications is susceptible to imbalanced class priors; instead, metrics such as AUC and f1-measure have been used because of their relative "immunity" to imbalanced class priors. As we have argued, application of these metrics to a problem where costs are supposedly "unknown" implies that hidden but known costs (equivalent to metric skew) are being used during evaluation. Neither AUC nor f1-measure have a skew ratio guaranteed to equal 1. F1-measure always has a higher skew for the positive class. For AUC, when $n^+ < n^-$ as in

[5] This is related to the problem with accuracy described in [Jap06]; using accuracy as an objective measure, however, each of these classifiers is equally good

imbalanced datasets, the skew ratio may often be greater than 1. For example, in the simple three-point ROC curve, the skew ratio is always equal to $\frac{n^-}{n^+}$ which, in an imbalanced dataset, is always greater than 1. Thus, the ability of AUC and f1-measure to effectively rank classifiers on imbalanced datasets seems to stem partially from the fact that the skew ratio favors the minority class, a bias which matches the property that the minority class has an implicit but unspecified misclassification cost greater than the misclassification cost on the majority class.

Other metrics also become more skewed towards the positive class as the prior of the negative class becomes larger than the prior of the positive class. In figure 4.9, we graph the indicator functions of whether the skew ratio is greater than 1; in the graphs, the color white indicates that the skew ratio is greater than 1 for those values of n_{tp} and n_{tn}, while black indicates that the skew ratio is less than or equal to 1. In the top row, we plot precision, while in the bottom row, we plot macroaveraged precision. The size of the negative class becomes larger with respect to the size of the positive class as one goes from left to right in the figure. Note that as the relative size of the negative class increases, the relative amount of space in the figures where the skew ratio is biased towards the positive class becomes larger and larger. In addition, only the right-most column has a positive class prior less than or equal to 0.1. That is, even when the negative class is slightly larger with respect to the positive class, the skew ratio favors the positive class for most possible values of n_{tp} and n_{tn}.

4.4 Relationship Between Misclassification Costs and Evaluation Metrics

Given a cost c_{fn} of making a false negative, a cost c_{fp} of making a false positive, and no costs for making a true positive or true negative, one can use cost-sensitive

learning to build a classifier which minimizes the total cost metric $c_{fn}n_{fn} + c_{fp}n_{fp}$. As discussed, one can show that the skew of this metric towards the positive class is $-c_{fn}$ while the skew of this metric towards the negative class is $-c_{fp}$. This suggests that there is a relationship between cost-sensitive learning and evaluation metric skews. Here, we will show that this relationship can be exactly defined, and that this viewpoint of metric skew and cost-sensitive learning is practically useful.

Given an evaluation metric expressed as a function $\boldsymbol{m}(n_{tp}, n_{tn})$, one can find a linear approximation to a function by using the first terms in a Taylor expansion. Let $\hat{\boldsymbol{m}}(n_{tp}, n_{tn})$ denote a linear approximation to some evaluation metric $\boldsymbol{m}(n_{tp}, n_{tn})$. Then, using just the linear terms in a Taylor expansion, one can define a linear approximation as follows:

$$\hat{\boldsymbol{m}}(n_{tp}, n_{tn}) = \boldsymbol{m}(n'_{tp}, n'_{tn}) + \frac{\partial \mathbf{m}}{\partial n_{tp}}(n_{tp} - n'_{tp}) + \frac{\partial \mathbf{m}}{\partial n_{tn}}(n_{tn} - n'_{tn}) \tag{4.13}$$

where n'_{tp} and n'_{tn} are the estimates of n_{tp} and n_{tn} where the Taylor approximation is being expanded. Note the constant terms in equation 4.13. Finding the values (n_{tp}, n_{tn}) that minimize $\hat{\boldsymbol{m}}(n_{tp}, n_{tn})$ is therefore equivalent to finding the values (n_{tp}, n_{tn}) that minimize

$$\hat{\boldsymbol{m}}_c(n_{tp}, n_{tn}) = \frac{\partial \mathbf{m}}{\partial n_{tp}} n_{tp} + \frac{\partial \mathbf{m}}{\partial n_{tn}} n_{tn} \tag{4.14}$$

Note the similarity to total misclassification cost, which is defined as $c_{fn}n_{tp} + c_{fp}n_{tn}$. Thus, one can optimize $\hat{\boldsymbol{m}}_c(n_{tp}, n_{tn})$, which is equivalent to optimizing the linear approximation to some evaluation metric $\boldsymbol{m}(n_{tp}, n_{tn})$, by using a cost-sensitive learner and setting $c_{fn} = \frac{\partial \mathbf{m}}{\partial n_{tp}}$ and $c_{fp} = \frac{\partial \mathbf{m}}{\partial n_{tn}}$. Of course, unless $\boldsymbol{m}(n_{tp}, n_{tn})$ is linear, $\hat{\boldsymbol{m}}(n_{tp}, n_{tn})$ will only be accurate around (n'_{tp}, n'_{tn}). We will address this in the next section.

In [Elk01], it is shown that a classifier that minimizes $c_{fn}n_{tp} + c_{fp}n_{tn}$ also

minimizes $\frac{c_{fn}}{c_{fp}} n_{tp} + n_{tn}$. That is, only the ratio of c_{fn} and c_{fp} is important when learning a cost-sensitive classifier. Thus, a classifier which minimizes equation 4.14 will also minimize $c_m n_{tp} + n_{tn}$, where we define c_m as follows:

$$c_m = \frac{\frac{\partial \mathbf{m}}{\partial n_{tp}}}{\frac{\partial \mathbf{m}}{\partial n_{tn}}} \tag{4.15}$$

Note that c_m is exactly equal to the skew ratio of the metric defined in the previous section. Thus, by using the metric skew of an evaluation metric as a linear approximation to an evaluation metric, one can use a cost-sensitive learner to optimize any evaluation metric which has a defined metric skew.

Interestingly, some empirical work has already pointed to this connection. In [WP03], the authors empirically show for various two-class problems that one can rebalance the priors in the training set in order to create classifiers that optimize either AUC or accuracy. The authors show that the general "guideline" is to use the natural priors when evaluating with accuracy and to use balanced priors when using AUC. In [Elk01], it is shown that one way to create a cost-sensitive classifier with a probabilistic classifier which labels a point as a particular class if the probability of coming from that class is greater than 0.5 is to rebalance the number of points in the training set such that the number of points in the positive class is equal to n^+ points and the number of points in the negative class is equal to $n^- \frac{c_{fp}}{c_{fn}}$ points. That is, there should be a ratio of n^+ positive class points to $n^- \frac{c_{fp}}{c_{fn}}$ points in the training set. This means that the ratio of positive to negative class points when optimizing for accuracy (which has skew ratio of 1) should be n^+ to n^-. That is, the natural priors are used. Similarly, when optimizing for AUC (which has skew ratio of $\frac{n^-}{n^+}$), the ratio of positive to negative class points should be 1 to 1. That is, balanced priors should be used. This is exactly consistent with the results in [WP03] and can be explained by the derivation above.

4.5 Active Learning for Handling Arbitrary Evaluation Metrics

In this section, we introduce a method for allowing uncertainty sampling to handle evaluation metrics besides total misclassification cost and 0-1 loss that builds on the work presented thus far in this chapter. Our proposed algorithm consists of two major components: 1) cost-sensitive uncertainty sampling with self training, and 2) metric skew.

4.5.1 Uncertainty Sampling for Arbitrary Evaluation Metrics

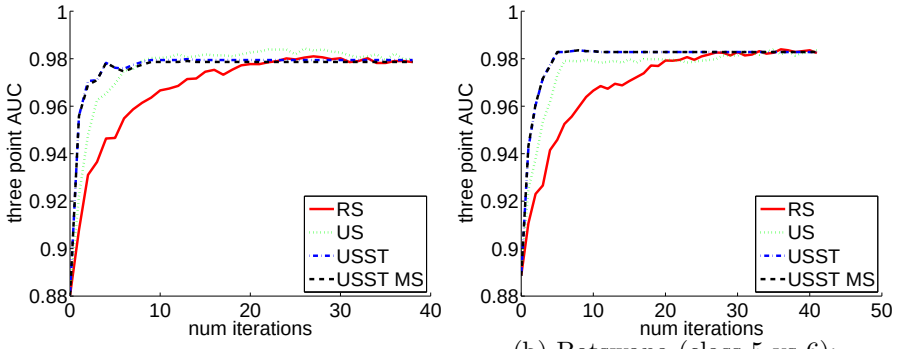
The proposed approach can be broken into the following basic steps on each iteration of active learning: retraining a classifier, finding a linear approximation for the desired evaluation metric, and then learning a cost-sensitive classifier based on this linear approximation.

The algorithm is given below.

1. **I**nput: Initial labeled set $\mathcal{L}$, unlabeled set $\mathcal{U}$, initial estimate of metric skew $c_m = 1$
2. On each iteration of active learning, do the following:
   (a) Select n points from $\mathcal{U}$ based on uncertainty sampling; label and add to $\mathcal{L}$
   (b) Retrain classifier on $\mathcal{L}$
   (c) Classify all points in $\mathcal{U}$; estimate values for n_{tp}, n_{tn}, n_-, and n_+
   (d) Calculate metric skew c_m
   (e) Using known labels and points in $\mathcal{L}$, predicted labels for $\mathcal{U}$, and current estimates of metric skew, train a cost-sensitive classifier where $c_+ = c_m$ and $c_- = 1$

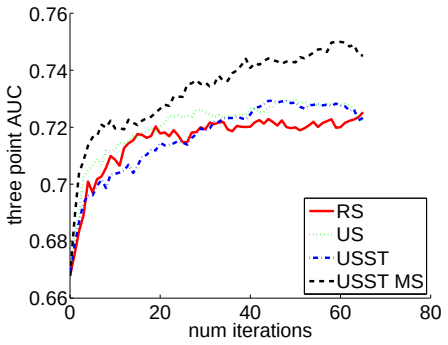
The first step (i.e., step 2a) on each iteration of active learning is the selection of points one would perform using "normal" uncertainty sampling where any previously proposed notions of uncertainty can be used (e.g., pick points with smallest margin between class with largest posterior probability and class with second largest posterior probability). The second and third steps on each iteration (i.e., steps 2b and 2c) are just the self-training steps in uncertainty sampling with self-training. The fourth step (step 2d) is the calculation of metric skew. Here, we calculate metric skew as described in the previous section, where values for n_{tp}, n_{tn}, n_-, and n_+ are calculated by splitting $\mathcal{LUU}$ into a training and validation set. Preliminary experiments indicate that the ratio of points in the training and validation sets does not affect results much; in the results presented, we use 90% of the self-trained set $\mathcal{LUU}$ for training a classifier to estimate metric skew and the remainder as a validation set. Note that self-training has two advantages in this algorithm: the creation of larger labeled validation and training sets, and the ability to create more accurate posterior probabilities. The final step is to use a cost-sensitive classifier with $c_+ = c_m$ and $c_- = 1$.

In summary, the proposed algorithm is a combination of cost-sensitive uncertainty sampling with self-training and the concept of metric skew given in the previous two sections, where the estimation of metric skew is intermingled with the selection of unlabeled points and the refinement of the training set. The idea of using metric skew as a cost and training a cost-sensitive classifier in order to optimize an arbitrary evaluation metric can be used independently of active learning. However, initial experiments show that in some cases, iterative estimates of metric skew are often more effective than simply estimating metric skew once. Thus, metric skew fits more naturally into the iterative refinement approach of typical active learning methods.

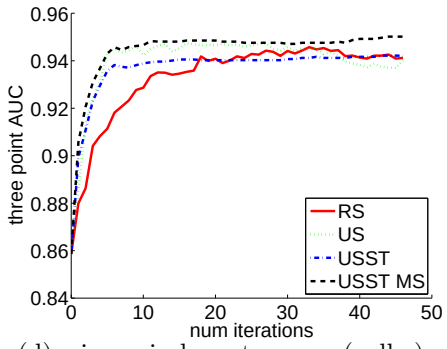


(a) KSC (class 3 vs 4); $\hat{c}_m = 0.959$

(b) Botswana (class 5 vs 6); $\hat{c}_m = 0.997$



(c) Pima Indian; $\hat{c}_m = 2.099$



(d) wisconsin breast cancer (wdbc); $\hat{c}_m = 1.705$

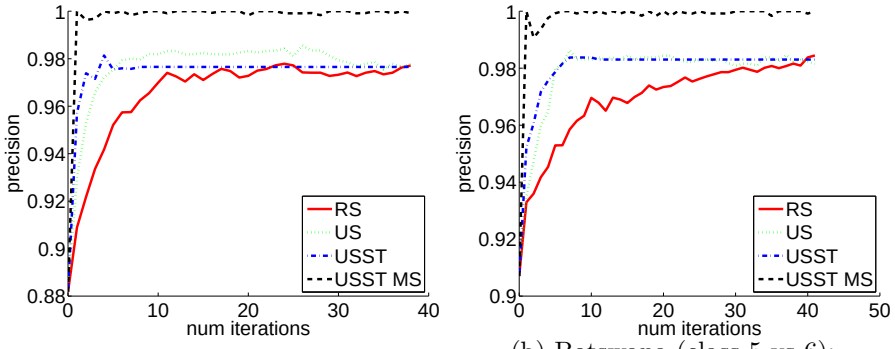
Figure 4.10: Sample results for LDA; three point AUC

4.5.2 Experiments

Experimental Setup

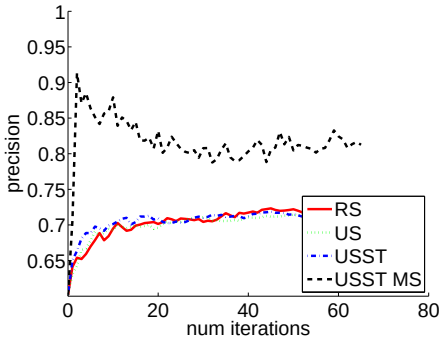
We present sample results on the hyperspectral data we used earlier in this chapter when running experiments for cost-sensitive active learners as well as on some low-dimensional datasets as described in Chapter 2.

In each of our experiments, we partition the data into training and test sets using five runs of ten-fold cross-validation and average the results. We use stratified sampling such that each fold has the same proportion of points from each class. Once the training and test sets have been created, 10% of the training data is randomly selected and labeled to form the initial labeled set $\mathcal{L}$, and the remaining unlabeled

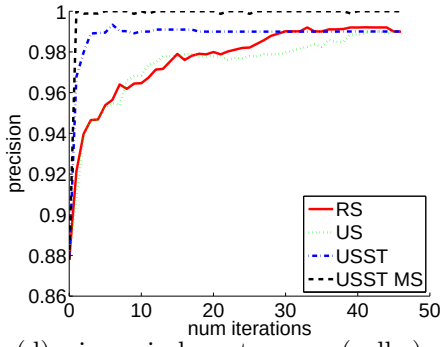


(a) KSC (class 3 vs 4); $\hat{c}_m = 0.150$

(b) Botswana (class 5 vs 6); $\hat{c}_m = 0.0293$



(c) Pima Indian; $\hat{c}_m = 0.266$

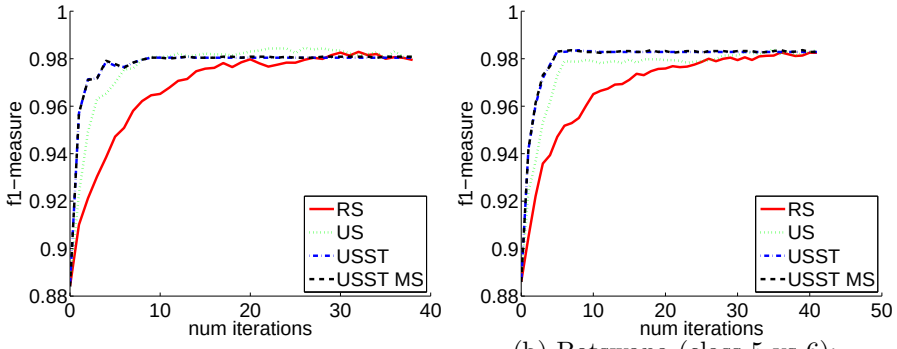


(d) wisconsin breast cancer (wdbc); $\hat{c}_m = 0.021$

Figure 4.11: Sample results for LDA; precision

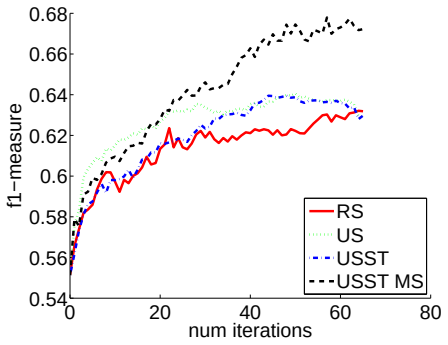
training data is placed in $\mathcal{U}$. On each iteration of active learning, 10 points are labeled. A different evaluation metric is used in each different experiment. We show results for three point AUC, precision, f1-measure, and geometric mean between precision and recall. Our proposed approach that combines metric skew with uncertainty sampling generalizes our proposed cost-sensitive uncertainty sampling with self training presented earlier in this chapter, so results if we are using total misclassification cost or accuracy are omitted from this section.

For the sake of variety, we also test our methods on three artificial evaluation metrics which, to the best of our knowledge, have not been used in other studies. We will simply call these artificial evaluation metrics "metric 1", "metric 2", and "metric 3". They are defined as follows: metric 1 $= n_{tp}^2 + n_{tn}^2$, metric 2$= n_{tp}^2 + n_{tn}$,

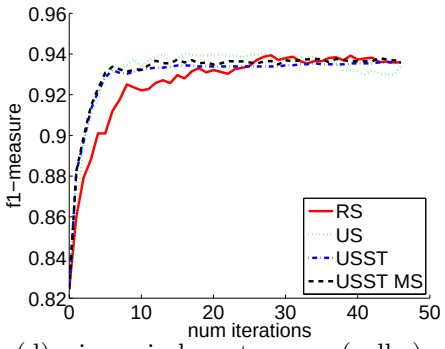


(a) KSC (class 3 vs 4); $\hat{c}_m = 1.044$

(b) Botswana (class 5 vs 6); $\hat{c}_m = 1.046$



(c) Pima Indian; $\hat{c}_m = 1.866$



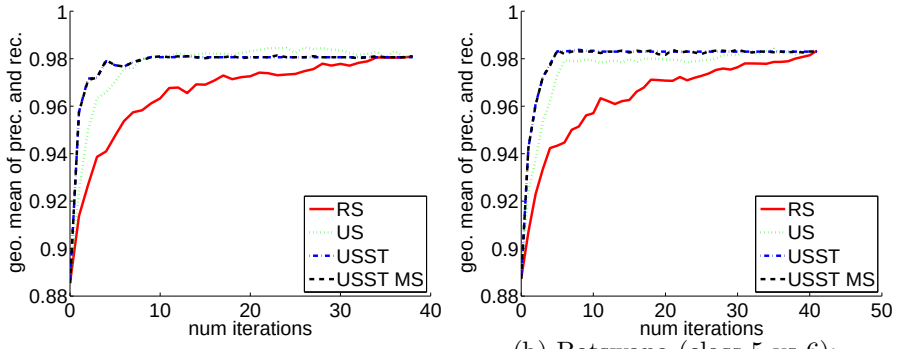
(d) wisconsin breast cancer (wdbc); $\hat{c}_m = 1.135$

Figure 4.12: Sample results for LDA; f1-measure

and metric 3= $n_{tp} * n_{tn}$. The purpose of including these evaluation metrics is not because we feel they are necessarily useful for comparing different classifiers, but simply to provide more evidence that our proposed approach works well.

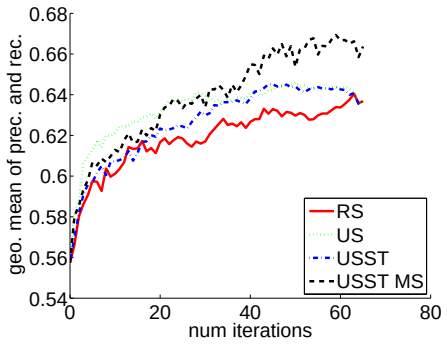
In each experiment, we use random sampling (RS), uncertainty sampling (US), and uncertainty sampling with self-training (USST) as baseline methods. We will refer to the proposed algorithm as the uncertainty sampling with self-training and metric skew approach, or "USST MS" for short.

We use an LDA classifier in all our experiments. We use uncertainty scores u_i equal to the inverse of the margin between the two posterior probabilities predicted for a point $\mathbf{x_i}$.

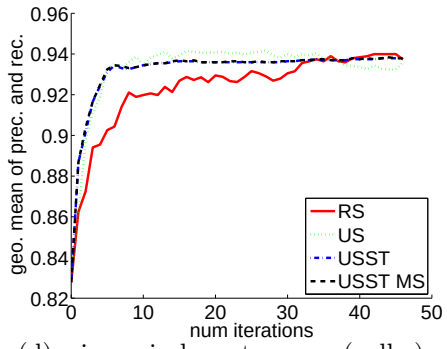


(a) KSC (class 3 vs 4); $\hat{c}_m = 1.048$

(b) Botswana (class 5 vs 6); $\hat{c}_m = 1.066$



(c) Pima Indian; $\hat{c}_m = 1.522$



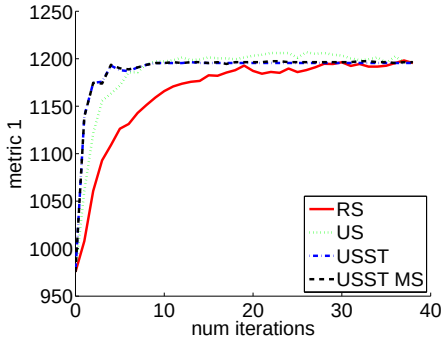
(d) wisconsin breast cancer (wdbc); $\hat{c}_m = 1.032$

Figure 4.13: Sample results for LDA; geometric mean of precision and recall

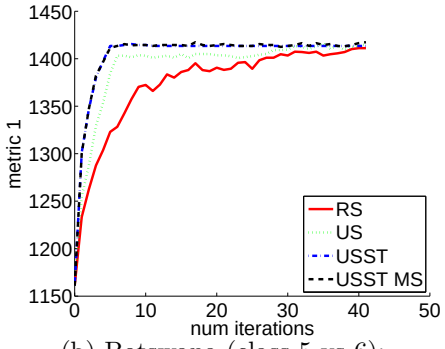
4.5.3 Results

In figures 4.10 through 4.16, we show some example curves on some of our datasets comparing the active learning curves of random sampling (RS), uncertainty sampling (US), uncertainty sampling with self-training (USST), and uncertainty sampling with self training and metric skew (USST MS).

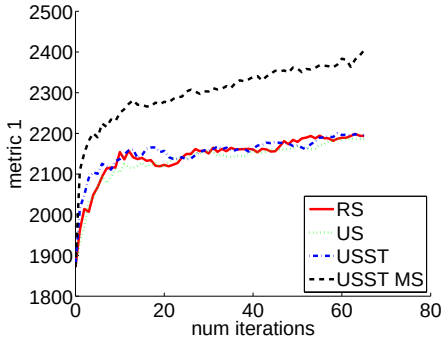
In the example curves, USST and USST MS tend to outperform the other methods. However, in practice, there seem to be many cases where USST and USST MS perform similarly. This appears to be because of the value obtained for c_m. In fact, in many cases, the estimated value for c_m is close to 1. For example, in cases where, say, $c_m \approx 1.2$, it is not surprising that USST and USST MS perform similarly.



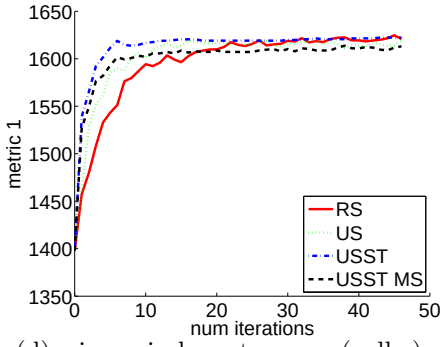
(a) KSC (class 3 vs 4); $\hat{c}_m = 1.057$



(b) Botswana (class 5 vs 6); $\hat{c}_m = 0.860$



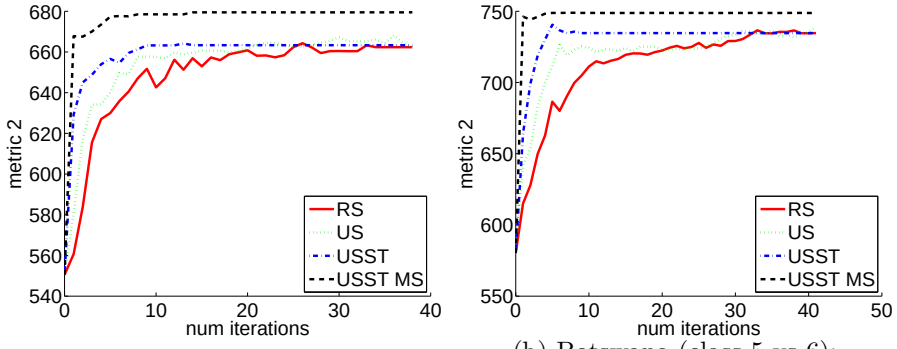
(c) Pima Indian; $\hat{c}_m = 0.355$



(d) wisconsin breast cancer (wdbc); $\hat{c}_m = 0.525$

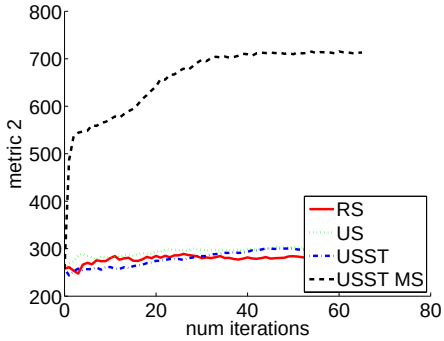
Figure 4.14: Sample results for LDA; metric 1

Thus, in the figures, we also list the average skew ratio $\hat{c}_m$ in the captions. When $\hat{c}_m$ is quite different from 1 (e.g., when the evaluation metric is precision or metric 2), USST MS tends to outperform USST. $\hat{c}_m$ also seems to differ more on many of the low-dimensional datasets we tested. Thus, we see more of a difference between USST MS and USST for many of these low-dimensional datasets. However, even if $\hat{c}_m$ is close to 1, USST MS never performs much worse than USST. Given that c_m is also simple to calculate, USST MS appears to be useful in practice compared to USST, regardless of whether c_m is close to or very different than 1.

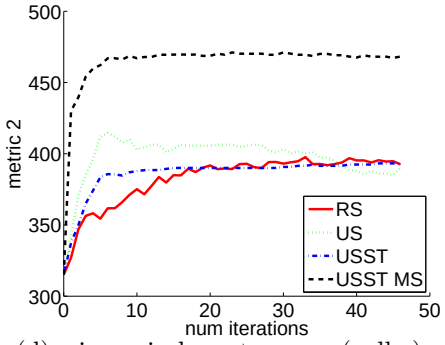


(a) KSC (class 3 vs 4); $\hat{c}_m = 45.421$

(b) Botswana (class 5 vs 6); $\hat{c}_m = 47.273$



(c) Pima Indian; $\hat{c}_m = 30.708$

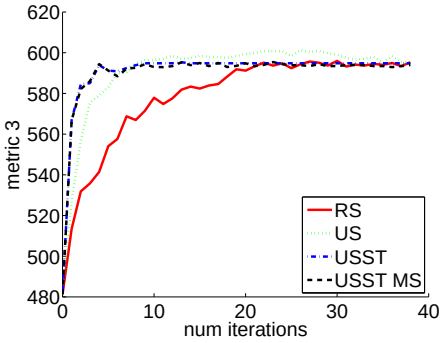


(d) wisconsin breast cancer (wdbc); $\hat{c}_m = 33.217$

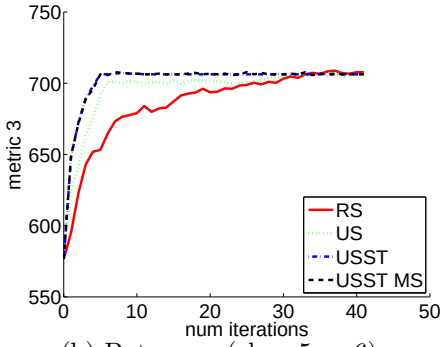
Figure 4.15: Sample results for LDA; metric 2

4.6 Conclusion

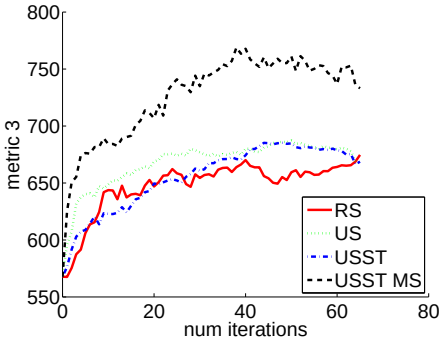
In this chapter, we have introduced several techniques. We first demonstrated that some form of semi-supervision is needed to perform cost-sensitive learning with uncertainty sampling, and proposed the use of self-training combined with uncertainty sampling. We then proposed a method for analyzing evaluation metrics which is related to previously proposed analysis frameworks and to cost-sensitive learning. In addition, we introduced a method for uncertainty sampling to handle any evaluation metric where metric skew can be defined. Empirical evidence supports our claim that the proposed approach is useful.



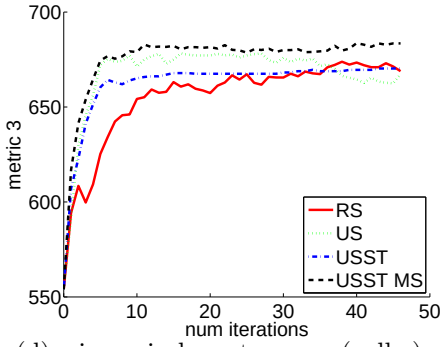
(a) KSC (class 3 vs 4); $\hat{c}_m = 0.938$



(b) Botswana (class 5 vs 6); $\hat{c}_m = 1.171$



(c) Pima Indian; $\hat{c}_m = 2.876$



(d) wisconsin breast cancer (wdbc); $\hat{c}_m = 1.922$

Figure 4.16: Sample results for LDA; metric 3

Chapter 5

Non-uniform Label Acquisition Costs in Active Learning

5.1 Introduction

In active learning, one attempts to maximize classifier performance for a given number of labeled training points by allowing the active learning algorithm to choose which points should be labeled. It is often assumed that: 1) the cost of acquiring the label for a particular point is independent of the label acquisition cost for all other points and that 2) label acquisition costs for all points are equal.

Costs that are non-uniform and are dependent on other points chosen for labeling may arise when applying active learning to spatially distributed data. For example, for classification of land-cover using hyperspectral data [Lan02], acquiring labels may involve traveling to a particular location and performing some sort of test such as determining the type of land at that point or collecting various samples, such as soil, water, or foliage samples, that require physical access. Traveling to this point incurs some type of cost (e.g., gas or time) proportional to the distance traveled. The distance traveled also depends on the order in which one visits the points that

need to be labeled, meaning that the label acquisition cost for a particular point is dependent on other, previously visited points.

In this chapter, we examine a method for incorporating non-uniform, dependent label acquisition costs into an uncertainty sampling [LC94] approach to active learning. We show that active learning with spatially dependent costs can be addressed by combining uncertainty sampling with techniques used to solve the traveling salesman problem (and its variant, the traveling salesman problem with profits). Finally, we show the relationship between the introduced method for handling spatial label acquisition costs and methods for handling non-uniform label acquisition costs in non-spatial data.

5.2 Spatially Dependent Label Acquisition Costs

5.2.1 Problem Setting

As in "standard" active learning, active learning on spatial data occurs in an iterative fashion where, on each iteration i, points from some unlabeled set $\mathcal{U}$ are selected by the active learner based on some criteria, labeled by some oracle, and then placed in the labeled set $\mathcal{L}$. As in the previous chapter, we focus on modifying an uncertainty sampling approach to active learning because of the computational efficiency of uncertainty sampling. In this chapter, we study two different but related scenarios for active learning on spatial data.

In the first scenario, on iteration i, the algorithm chooses a fixed number of points to label and provides the labeler a path to travel between the points, beginning and ending at the same "home location" on each iteration[1]. Once the labels have been obtained, the algorithm retrains on the set of labeled points $\mathcal{L}$, and the process repeats. The goal is to produce a classifier that reduces empirical errors

[1]This home location may correspond to where the labeler's vehicle is stored/refueled or the labeler's base of operations.

while minimizing data acquisition costs.

The second scenario also involves a "home location" where the labeler begins and ends on each iteration. However, instead of labeling a fixed number of points on each iteration, the labeler continues to label points until some traveling budget is expended. In this chapter, the traveling budget is the amount of time available for traveling, sample gathering and labeling each day.

For the sake of discussion, we will refer to the first scenario as spatial active learning with a fixed batch size and refer to the second scenario as spatial active learning with a fixed traveling budget. The two problems are essentially the same except for different constraints; the solutions to these two problems are therefore quite similar.

We will use the following notation. On the ith iteration, the algorithm selects n_i points for labeling. In the first scenario, n_i is constrained to be equal to a fixed batch size n_{fix} on each iteration. In the second scenario, n_i depends on some traveling budget which we will denote as t_{max}. The actual cost of traveling and labeling points for the ith iteration will be denoted as t_i. t_i depends on the total distance d_i traveled on the ith iteration, the speed s of the labeler's vehicle, the cost of labeling a single point c_l, and the number of points labeled n_i. In particular,

$$t_i = (d_i/s) + (c_l * n_i) \tag{5.1}$$

and the constraint is that $t_i < t_{max}$. We will measure t_i, t_{max}, and c_l in units of time, d_i in units of length, and s in units of length/time.

Finally, we will use uncertainty scores for each point in $\mathcal{U}$ as is done in uncertainty sampling. We will denote the uncertainty score for the jth point in $\mathcal{U}$ as $u(j)$.

5.2.2 Combining Active Learning with the Traveling Salesman Problem

A simple but somewhat naive approach for incorporating spatially related label acquisition costs with active learning to solve the scenario with fixed batch size n_{fix} is to do the following:

1. On each iteration, based on active learning criteria (e.g., uncertainty sampling or random point selection), select n_{fix} points to label
2. Use a solution to the traveling salesman problem (TSP) to find the shortest path from home through all of the n_{fix} selected points

We consider this method naive because all spatial information is ignored when choosing the initial n_{fix} points. While these n_{fix} points may be highly informative (e.g., as measured by their uncertainty scores), it may be less costly to select n_{fix} points which are closer to each other. We use this approach as a baseline for comparisons.

This baseline algorithm is somewhat clumsy to extend for the scenario where there is a fixed traveling budget since it is unknown how many points n_i to label on a particular iteration. In order to keep a similar baseline for both scenarios, we use this somewhat inefficient approach:

1. Based on the labeling budget t_{max}, estimate the maximum number of points n_{max} that could be labeled per iteration if one ignores traveling costs; that is, let $n_{max} = t_{max}/c_l$
2. Set $n_i = n_{max}$
3. Find the total cost t_i for labeling all n_i points while traveling along an optimal path (as found using a solution to TSP) through all n_i points, starting and ending at home

4. If $t_i > t_{max}$, set $n_i = n_i - 1$ and repeat previous step; otherwise, ask oracle for labels of current n_i points

We will refer to this baseline algorithm as "random/TSP" if random sampling is used to select points or "US/TSP" if uncertainty sampling is used to select points.

5.2.3 Combining Active Learning with Traveling Salesman with Profits.

The traveling salesman problem with profits (TSPP) is a variant of TSP. In TSPP, each city that the salesman can travel to has a profit associated with visiting that city, and the salesman is not required to visit every city. The goal is to find an optimal path through some subset of the cities that maximizes total profit under some constraint.

Our approach for performing active learning on spatial data is to transform the problem into a traveling salesman problem with profits. On each iteration of active learning, the cities that can potentially be visited by the salesman (or labeler) are the points in the unlabeled set $\mathcal{U}$. The profit associated with visiting an unlabeled point is set equal to the uncertainty score of that particular unlabeled point. That is, $p(j) = u(j)$, where $p(j)$ is the profit for visiting the jth unlabeled point and $u(j)$ is the uncertainty score of the jth unlabeled point.

The constraint under which the salesman must travel differs for our two scenarios of fixed batch size and fixed traveling budget per iteration. In the fixed batch size scenario, the constraint is that the salesman can only visit n_{fix} points per iteration. In the fixed traveling budget scenario, the salesman can visit a variable number of cities per iteration as long as the total time required to travel along all cities and reach home is less than the traveling budget t_{max} for that iteration.

The traveling salesman with profits has been studied extensively [FDG05], and fits quite well into the problem of active learning with spatial costs. We refer

to this approach as "US/TSPP".

5.2.4 A Generalization of Active Learning with Traveling Salesman with Profits

Empirically, we found a variant of US/TSPP to be useful: instead of supplying all possible unlabeled points to the traveling salesman with profits algorithm, only the top m points with the highest uncertainty scores (where $m \geq n_i$) are used. The advantages of this variation on our framework is that it is computationally more efficient and does well at trading off between maximizing classification performance and minimizing distance traveled. We refer to this approach as "US/TSPP (filtered)".

More formally, our approach is as follows:

1. Input: number of points to consider m, labeled set $\mathcal{L}$, unlabeled set $\mathcal{U}$, constraint n_{fix} (for scenario with fixed batch size) or t_{max} (for scenario with fixed traveling budget)

2. Iterate

   (a) Assign uncertainty scores $\mathbf{u}$ to all points in $\mathcal{U}$

   (b) Select the min(m, $|\mathcal{U}|$) points with the highest uncertainty scores where $|\mathcal{U}|$ is the number of points in $\mathcal{U}$; let this set of unlabeled candidate points be denoted as $\mathcal{U}_C$

   (c) Set the profit for visiting the jth point in $\mathcal{U}_C$ to the uncertainty score of that point

   (d) Select points from $\mathcal{U}_C$ and create a path through these points using a solution to TSPP that satisfies traveling constraints (either n_{fix} or t_{max} depending on the scenario)

For the scenario with fixed batch size, this method generalizes both US/TSP

and US/TSPP. Note that if $m = n_{fix}$, then this approach reduces to US/TSP, while if m is equal to the number of points in $\mathcal{U}$, then this approach reduces to US/TSPP.

5.3 Experiments

5.3.1 Solving TSP and TSPP

In this section, we discuss heuristics for solving the traveling salesman problem and traveling salesman problem with profits in more detail. These specific algorithms are necessary for understanding our experiments. However, our solutions do not require a specific solution to TSP or TSPP, so other approaches can be readily substituted within the framework presented in this chapter.

The traveling salesman problem has been widely studied[LLKS85], and many solutions have been proposed. We will use the following terminology and notation. Each city that the salesman travels to is denoted as a node $v(i)$, where $v(i)$ is the ith city. In addition, the salesman must start and end at some home location $v(0)$. Finally, the distance between two nodes i and j is denoted by $\boldsymbol{d}(i, j)$, where $\boldsymbol{d}(0, j)$ would represent the distance from the "home" location to the jth point.

We use the following heuristic, chosen because the solution is quite fast and also because of readily available code[2]. Our heuristic for solving the traveling salesman problem works as follows. The algorithm begins by randomly initializing a path through all points. The algorithm then proceeds to try to improve the path by repeatedly using 2-opt and 2.5-opt, where 2-opt refers to exchanging the position of two nodes in the current path, and 2.5-opt refers to removing a single node from the current path and inserting it between two existing nodes in the path. One problem with this heuristic is determining when to terminate the algorithm. We found through preliminary experiments that setting the max number of iterations

[2]Our experiments were run using a modified version of the code used in the Matlab demonstration of the traveling salesman problem (travel.m)

to 10,000 was quite fast for a moderate number of nodes (i.e., less than 100) and resulted in convergence to a good solution.

We use a variant of the H2 heuristic originally presented in [GLS98] to solve the traveling salesman with profits problem. Our modified algorithm is as follows:

1. Initialization: initialize path to consist of $v(0)$ (i.e., the home node) and the point j that minimizes $\frac{\boldsymbol{d}(0,j)+\boldsymbol{d}(j,0)}{p(j)}$ where $p(j)$ is the profit obtained for visiting node j

2. Iterate until the number of nodes (not counting $v(0)$) in the path is equal to n_{fix} (fixed batch size constraint) or until the total traveling budget exceeds t_{max} (fixed budget constraint): Add point that minimizes $\frac{\boldsymbol{d}(i,j)+\boldsymbol{d}(j,k)-\boldsymbol{d}(i,k)}{p(j)}$ for some point $v(j)$ not in current path and consecutive points $v(i)$ and $v(k)$ in path

3. Optimize path using solution to TSP described above

As with our method for solving the traveling salesman problem, this algorithm was chosen since it was simple to implement and because it was fast enough for experimentation.

5.3.2 Classifiers

As mentioned, we use two classifiers in our experiments: LDA and SVMs. These classifiers were chosen for experimentation with spatial label acquisition costs because they performed well on "traditional" active learning with non-spatial label acquisition costs on hyperspectral data. For LDA, we use uncertainty sampling where the uncertainty scores are inversely proportional to the margin between the largest posterior probability and second largest posterior probability as predicted by LDA. For SVMs, the uncertainty score is inversely proportional to the absolute

value of the distance from the hyperplane, where points closer to the hyperplane are more uncertain [TK02].

For SVMs, we found that a linear kernel worked well; moreover, there are fewer parameters to be tuned for a linear kernel. One parameter, however, is the trade-off between margin and empirical error on the training set. We found that setting this parameter equal to one seemed to work well. However, the problem of tuning parameters for SVMs during active learning is, to the best of our knowledge, still an open question. Typically, in classification, one would tune parameters using ten-fold cross validation or by using a single hold-out set for validation. This may not work well in active learning for two reasons: speed and lack of data. Because of the large number of iterations required in active learning, retuning the parameters for each iteration may be unwieldy, particularly if one were to use some form of cross-validation. In addition, because of the lack of data, particularly in early iterations of active learning, there is typically not enough data to reserve a separate hold-out set. In order to avoid confounding the results of the current study with any affects due to some form of dynamic parameter estimation, we use as few parameters as possible (i.e., we choose a linear kernel) and keep all other parameters constant (i.e., we set trade-off between margin and empirical error on the training set equal to one).

5.3.3 Datasets

As use the KSC and Botswana hyperspectral datasets described in chapter 2. In each dataset, each data point has a location as determined by two coordinates. We use the Euclidean distance between coordinates as the distance required to travel between two points.

We study both the full multiclass problem as well as several two-class problems. We used only LDA on the multiclass problem, and used both LDA and SVM on the two-class problems. The two-class problems were selected such that the fol-

lowing question could be answered: for a two-class problem where it is known that "traditional" active learning worked well, how much better does a technique that takes spatial information into account perform?

5.3.4 Experimental Setup

We partition the data into training and test sets using five runs of ten-fold cross-validation and average the results. In particular, we use stratified sampling such that each fold has the same proportion of points from each class.

Once the training and test sets have been created, 10%[3] of the training data is randomly selected and labeled to form the initial labeled set $\mathcal{L}$, and the remaining unlabeled training data is placed in $\mathcal{U}$.

We use the best-basis feature reduction method to create a set of d' features. The value of d' was chosen such that $|\mathcal{L}| > d'$ since this is required by LDA to form a covariance matrix. Experimentally, this was done by setting d' to the smallest value in the set of $(40, 50, 60)$ that would satisfy the constraint $|\mathcal{L}| > d'$ if 10% of the training data were placed in $\mathcal{L}$[4].

We run two sets of experiments. In the first set of experiments, we address the scenario where a fixed number of points are labeled on each iteration. In these experiments, the number of points n_{fix} labeled on each iteration is set to 10 points, and labeling costs are measured in terms of distance. In the second set of experiments, we address the scenario with fixed traveling budget. In this scenario, labeling costs are measured in units of time, where the constraint on each iteration is that the time t_i spent traveling and labeling points on the ith iteration is less than t_{max}. As mentioned, t_i depends on the vehicles average speed s and the cost for labeling a single point c_l. Experimentally, we tried a variety of values for s and c_l, but

[3]For multiclass datasets only, we used 5% of the data as seeds due to the larger size of the dataset

[4]Because of the larger dataset size in the multiclass experiments, it was possible to set d' equal to 120

found that specific values do not seem to affect general trends much. Because of space constraints, we present results only for the case where $s = 80.5$ kilometers per hour[5], $c_l = 10$ minutes, and $t_{max} = 8$ hours. Finally, US/TSPP(filtered) requires a parameter m to determine the number of points in $\mathcal{U}_C$, which we simply set to 100 in all experiments. Although we did not investigate this, additional tuning of m may be useful.

5.4 Results

Table 5.1: Averaged values for specific points on our graphs for scenario with fixed batch size (top two tables) and scenario with fixed traveling budget (bottom two tables)

dataset	activeLearner	errorRateAt50KM	errorRateAt100KM	errorRateAt250KM
ksc	random/TSP	0.1231	0.1143	0.1000
ksc	US/TSP	0.1114	0.0954	0.0688
ksc	US/TSPP(filt)	0.1032	0.0870	0.0606
ksc	US/TSPP	0.0974	0.0868	0.0701
bots	random/TSP	0.2555	0.2022	0.1215
bots	US/TSP	0.2447	0.1852	0.0970
bots	US/TSPP(filt)	0.1471	0.0990	0.0513
bots	US/TSPP	0.0947	0.0733	0.0521

dataset	activeLearner	errorRateAt25 hours	errorRateAt50 hours	errorRateAt100 hours
ksc	random/TSP	0.1137	0.1015	0.0911
ksc	US/TSP	0.0956	0.0734	0.0515
ksc	US/TSPP(filt)	0.0886	0.0655	0.0465
ksc	US/TSPP	0.0912	0.0752	0.0592
bots	random/TSP	0.1353	0.1268	0.0783
bots	US/TSP	0.1292	0.1109	0.0557
bots	US/TSPP(filt)	0.1150	0.0689	0.0363
bots	US/TSPP	0.0760	0.0590	0.0450

Multiclass results are presented in figure 5.1 and table 5.1. For both scenarios (fixed batch size and fixed traveling budget), the general trends and shapes of the

[5]i.e., roughly 50 miles per hour.

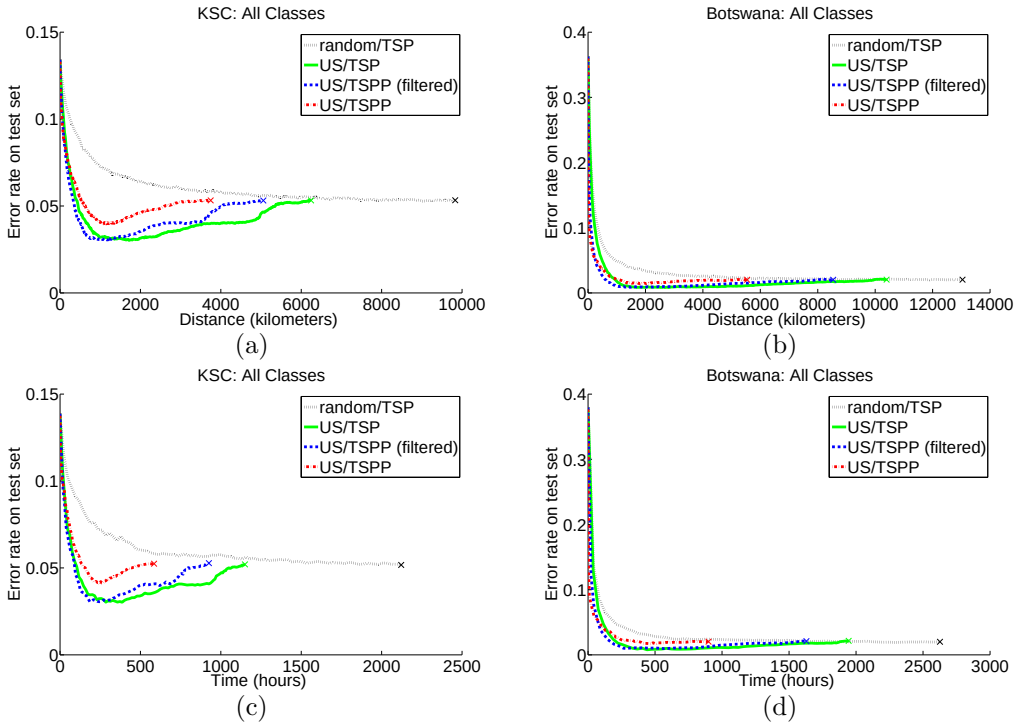
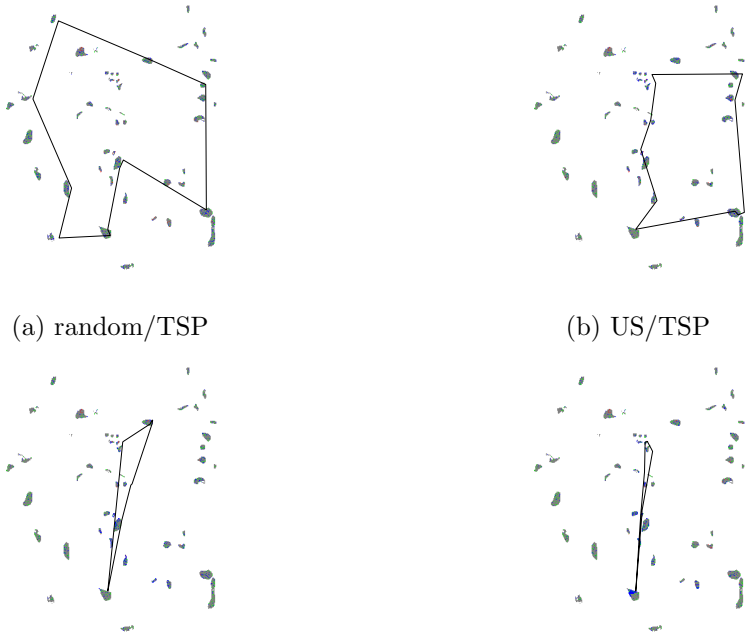


Figure 5.1: Average results; top row contains results for scenario I (fixed number of points per iteration) while bottom row contains results for scenario II (fixed budget per iteration)

curves in the results are quite similar.

In the graphs in figure 5.1, we have modified the traditional "banana curve" format of active learning in order to take into account different label acquisition costs. In this case, the horizontal axis is proportional to the total cost (e.g., distance traveled or time expended) needed to label $\mathcal{L}$. Thus, a point on the curve represents the cost expended by a labeler in order for the classifier to achieve a certain level of performance as measured by error rate on the test set. A number of interesting observations can be drawn from this type of figure. Unlike a traditional curve in active learning, the right-most point of each curve on each graph will be at a different place. This point corresponds to where all the points originally in $\mathcal{U}$ have been labeled (we have denoted this point with an "X" in the graphs for ease of

visibility). The fact that different curves end at different places indicates that each method requires different amounts of effort to label all of $\mathcal{U}$.



(a) random/TSP

(b) US/TSP

(c) US/TSPP (filtered)

(d) US/TSPP

Figure 5.2: Example paths for each of the methods in our experiments

Based on this right-most point, in terms of minimizing the total cost to label all of $\mathcal{U}$, US/TSPP outperforms US/TSPP (filtered) which outperforms US/TSP which outperforms random sampling on both datasets for both scenarios. This shows that our proposed approach, US/TSPP always reduces total labeling effort compared to US/TSP when labeling all of $\mathcal{U}$, and that US/TSPP (filtered) lies somewhere in-between. In many cases, this reduction is quite large. For example, the cost for labeling all of $\mathcal{U}$ using US/TSPP is, on average, less than half the cost if using US/TSP. For the sake of illustration, we have plotted four example paths on the KSC dataset[6], with one path for each tested method, in figure 5.2.

[6]we arbitrarily chose to plot the 25th iteration of active learning for each method for the fixed

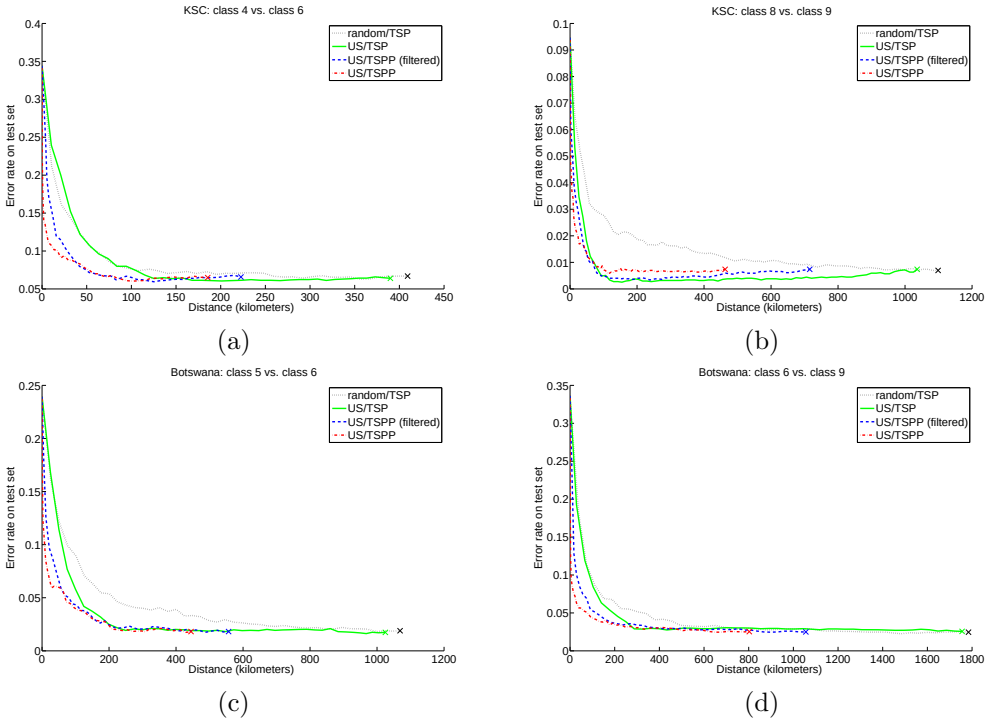


Figure 5.3: Sample results for two class problems using LDA in Scenario I

Here, the distance saved by US/TSPP and US/TSPP(filtered) over random/TSP and US/TSP is also apparent.

Another question in active learning is how quickly a method reduces classification error. Looking at the graphs, one can qualitatively make the judgement that US/TSPP tends to reduce the error rate fastest. Quantitatively, one can compare the average error rate when the labeler has not yet traveled very far. In table 5.1, we have listed some error rates when the traveler has not expended much cost. If looking at the lowest obtainable error rate, US/TSPP(filtered) is able to reduce error rate further than US/TSPP. In general, the lowest error rate (regardless of how much effort required to reach that point) is comparable for US/TSPP (filtered) and US/TSP. Thus, US/TSPP(filtered) empirically achieves an error rate as low as

batch size scenario.

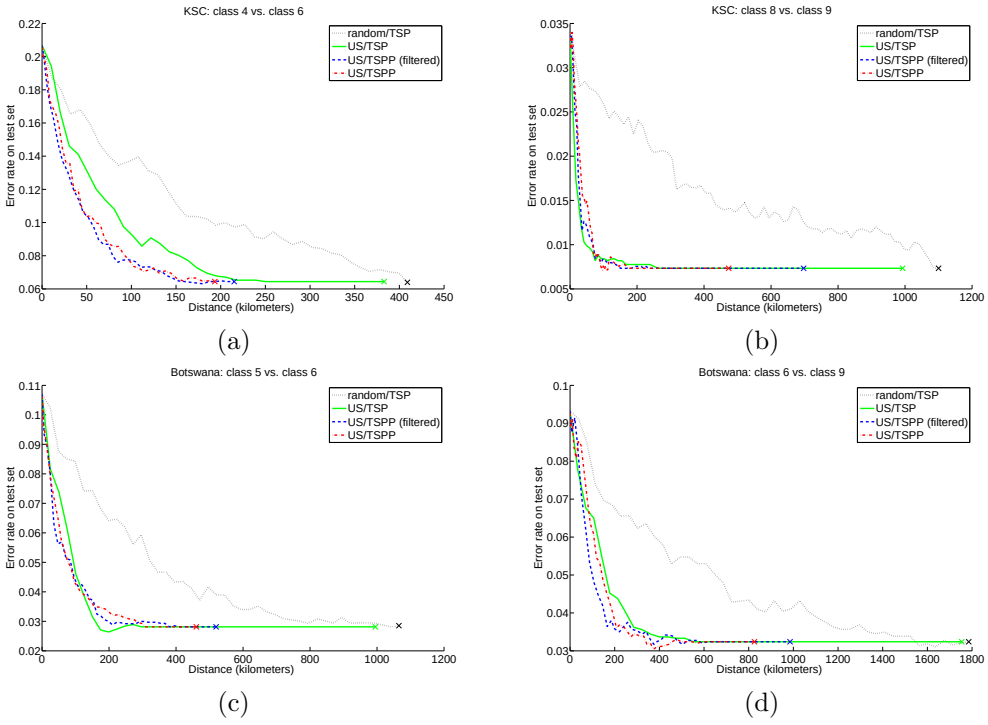


Figure 5.4: Sample results for two class problems using SVMs in Scenario I

US/TSP and reduces error rate more quickly than US/TSP.

Finally, in the results (particularly on the KSC dataset), we see that the error rate can be lower when only a subset of $\mathcal{U}$ has been labeled as opposed to when all of $\mathcal{U}$ has been labeled. This phenomenon has been observed in other works on active learning as well [SC00]. Stopping the labeling process early is useful in reducing overall classification error in the presence of outliers in the training set.

Sample results on two-class problems with both LDA and SVMs are shown in figures 5.3 and 5.4 for scenario I. The trends are similar to the multi-class problems.

5.5 Relationship to Non-spatial Label Acquisition Costs

Above, we described how spatially driven label acquisition costs can be combined with uncertainty sampling. In both [HSRC08] and [SCF08], the authors showed that one can combine label acquisition costs with uncertainty sampling for non-spatial applications. In particular, given that u_i is the uncertainty score of the ith point and l_i is the label acquistion cost, then one labels the point $\mathbf{x_i}$ with highest utility u_i/l_i. Let us call this the "utility-based framework for uncertainty sampling," or simply the "utility-based framework" for short.

One can show that the H2 algorithm used to solve the traveling salesman problem with profits is a greedy algorithm that will pick the same points as the utility-based framework. One can see the equivalence between the utility-based framework and the H2 algorithm when used to minimize spatial label acquisition costs by rewriting the utility-based framework as follows. Instead of picking n points with largest values u_i/l_i, we re-write the utility-based framework in an equivalent but alternate manner.

1. Initialization: Find point $\mathbf{x_i}$ that minimizes $\frac{l_i}{u_i}$; add $\mathbf{x_i}$ to set of points to label
2. Iterate until n points selected: Add point $\mathbf{x_i}$ that minimizes $\sum_j \frac{l_j}{u_j}$, where the summation is taken over all points already selected for labeling this iteration and the new point $\mathbf{x_i}$

Clearly, this gives the same points as the utility-based framework that looks at maximal values of u_i/l_i. In our framework for spatial label acquisition costs, we set p_j, the profit for visiting the jth point, equal to u_j, and we set the label acquisition cost for a batch of n points equal to the total distance traveled to reach those n points. If we let $\boldsymbol{d}(0,j) + \boldsymbol{d}(j,0)$ be the label acquisition cost of the first point labeled, and let $\boldsymbol{d}(i,j) + \boldsymbol{d}(j,k) - \boldsymbol{d}(i,k)$ be the label acquisition cost of all successive

points labeled on that same iteration of active learning, then it is clear that the H2 algorithm produces the same results as the utility-based framework.

This is particularly interesting given that all three approaches (our approach presented above and the approaches in [HSRC08] and [SCF08]) were not only published independently at the same time, but all three works were presented at the same venue. That is, it is interesting that all three approaches chose to modify an uncertainty sampling approach to handle label acquisition costs.

5.6 Conclusion

When performing active learning on spatially distributed data such as hyperspectral data or other GIS applications, there are variable label acquisition costs involved in labeling each point. These label acquisition costs may depend on the distance the labeler needs to travel in order to label each point selected by active learning. Standard active learning techniques are unable to handle such variable costs, and assume that the cost of labeling each point is uniform and that the cost of labeling each point is independent of all other points being labeled on a particular iteration of active learning. If the cost of labeling a point is proportional to the distance needed to travel to that point, and if the labeler travels to several points on each iteration of active learning, then both of these assumptions made by traditional active learning are broken.

In this chapter, we have presented methods for solving this variant of active learning where label acquisition costs are proportional to the distance required to travel to a point. We have addressed two scenarios for spatial active learning: one where there are a fixed number of points labeled on each iteration, and one where the number of points labeled on each iteration are dependent on some fixed budget.

Finally, we showed that our approach can be generalized to handle label-acquisition costs in non-spatial situations as well. Thus, our approach can be used in

any situation where one wants to apply uncertainty sampling and there are different label acquisition costs for each point.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This dissertation developed computationally efficient active learning methods that can account for different notions of utility. In order to do this, we first introduced generative oversampling, a parametric approach to resampling for imbalanced datasets. We then studied the relationship between random and generative oversampling in relationship to cost-sensitive learning. In particular, we furthered the currently known relationship between resampling and cost-sensitive learning, and looked at cases where resampling and cost-sensitive learning were and were not equivalent. This work on imbalanced datasets and cost-sensitive learning motivated the need for a general method for cost-sensitive active learning.

We also introduced a framework for analyzing evaluation metrics called metric skew. We show a number of situations where this framework can be useful in analyzing evaluation metrics, and made connections between metric skew and previously introduced methods for analyzing evaluation metrics. We also show that metric skew can be used to improve the capabilities of active learning.

Our work on active learning modifies uncertainty sampling, a computation-

ally efficient, empirically-driven approach to active learning. It allows uncertainty sampling to handle points with different misclassification costs, points with different label acquisition costs, and problems with different evaluation metrics. This is extremely important in practical applications of uncertainty sampling since problems with different misclassification costs or evaluation metrics are common, and we hope other researchers and practitioners will find the presented approaches useful.

6.2 Future Work

A number of possible areas of future work are possible.

Our theoretical analysis of generative oversampling and multinomial naive Bayes suggests a faster method of accomplishing the same effect as generative oversampling without the computational need to perform resampling. Since the expected value of the multinomial parameters after resampling can be derived exactly, one can simply use these values for the parameters in multinomial naive Bayes. Our experiments show that this is empirically superior to Laplace smoothing when running multinomial naive Bayes. Future work needs to compare this approach with other alternatives to Laplace smoothing. In addition, while this technique appears to be empirically superior, future work will hopefully show theoretical reasons why this approach is preferable.

In chapters 3 and 4, one motivation for cost-sensitive active learning is the connection between resampling and active learning. Another possibility is to create a framework that chooses between multiple oracles. For example, one oracle could be resampling (a fast but possibly untrustworthy oracle) and the other could be a human labeler (a slower but more trustworthy oracle). In general, the framework could be useful for other cases, such as in text classification, where one could envision several different sources of labeled information. For example, human labelers could label words versus documents, where labeling words might be faster but less

informative than labeling entire documents. One possible approach to create a general framework that can handle these different oracles is an n-armed bandit like framework.

In chapter 4, we show that self-training is useful for cost-sensitive learning. The metric we use to measure "usefulness" is, of course, misclassification cost. One possible avenue of future work is to directly study the effect of self-training on posterior probability estimation. This study should also compare results with bootstrap-lv and determine whether bootstrap-lv could benefit from a self-training approach.

In chapter 4, we introduce a method for analyzing evaluation metrics. Additional analysis and connections to other work can be made. For example, recently, in [Han09], limitations of the AUC metric are described. In short, it is shown that AUC is a method of calculating a weighted sum of the performance of a classifier over several costs. However, it is also shown that the weights used to calculate the AUC are classifier dependent. Thus, comparing AUC across several classifiers is extremely problematic. Metric skew appears to be an alternative method of explaining this phenomena, but future work needs to show the exact connection between metric skew and the work presented in [Han09].

Finally, one theme of the work in this dissertation is that many techniques are classifier and dataset dependent. For example, resampling has predominantly been studied with decision trees in other works, and we have shown that results obtained from resampling experiments differ when different classifiers are used. As another example, results on cost-sensitive uncertainty sampling show that self-training seems to be more useful for generative instead of discriminative classifiers. Thus, in future work, one could cover an even wider variety of classifiers.

Appendix A

Resampling

In this section, we will derive some of the equations found in Chapter 3, particularly those on oversampling for Gaussians and multinomials.

A.1 Gaussian Random Oversampling

Let the set of n points in the positive class be denoted as X_1, . . ., X_n. These points are a random sample from a normal distribution with true mean μ and true variance σ^2. Let us now consider a random sample $X_1^{(r)}, \ldots, X_m^{(r)}$ produced through random oversampling. Thus, each $X_j^{(r)}$ equals some X_{K_j}, where $K_1, \ldots, K_m$ are iid uniformly distributed random variables over the discrete set $\{1, \ldots, n\}$. The sample mean, $\bar{X}^{(r)}$, of only the randomly resampled data is an unbiased estimator of the true mean, since

$$E[\bar{X}^{(r)}] = \frac{1}{m}\sum_{j=1}^{m} E[X_j^{(r)}] = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} E[X_{K_j}|K_j = i]P[K_j = i] = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} \mu\frac{1}{n} = \mu, \tag{A.1}$$

noting that $E[X_{K_j}|K_j = i] = E[X_i] = \mu$ for all i and j. It follows that the sample mean, $\bar{X}_*^{(r)}$, of the pooled sample, $X_1, \ldots, X_n, X_1^{(r)}, \ldots, X_m^{(r)}$, is also unbiased, since

$$E[\bar{X}_*^{(r)}] = \frac{E[n\bar{X} + m\bar{X}^{(r)}]}{n+m} = \frac{nE[\bar{X}] + mE[\bar{X}^{(r)}]}{n+m} = \mu. \tag{A.2}$$

The variance of the sample mean for only the resampled data is determined as follows. First, note that

$$\mathrm{Var}[\bar{X}^{(r)}] = \frac{1}{m^2}\mathrm{Var}\left[\sum_{j=1}^{m} X_j^{(r)}\right] = \frac{1}{m^2}\left[\sum_{j=1}^{m}\mathrm{Var}[X_j^{(r)}] + \sum_{j\neq j'}\mathrm{Cov}[X_j^{(r)}, X_{j'}^{(r)}]\right] \tag{A.3}$$

Now, the variance of an individual sample is

$$\mathrm{Var}[X_j^{(r)}] = \sum_{i=1}^{n}\mathrm{Var}[X_{K_j}|K_j = i]P[K_j = i] = \sum_{i=1}^{n}\mathrm{Var}[X_i]\frac{1}{n} = \sigma^2, \tag{A.4}$$

while the covariance between different samples is

$$\mathrm{Cov}[X_j^{(r)}, X_{j'}^{(r)}] = \sum_{i,i'}\mathrm{Cov}[X_{K_j}, X_{K_{j'}}|K_j = i, K_{j'} = i']P[K_j = i, K_{j'} = i'] = \sum_{i,i'}\frac{\sigma^2\,\delta_{i,i'}}{n^2} = \frac{\sigma^2}{n}. \tag{A.5}$$

Thus,

$$\mathrm{Var}[\bar{X}^{(r)}] = \frac{1}{m^2}\left[m\sigma^2 + m(m-1)\frac{\sigma^2}{n}\right] = \left(1 + \frac{n-1}{m}\right)\frac{\sigma^2}{n}, \tag{A.6}$$

and we note that the variance is larger than that for the original sample (i.e., $\mathrm{Var}[\bar{X}] = \sigma^2/n$).

If the data is pooled, the variance of the sample mean becomes

$$\mathrm{Var}[\bar{X}_*^{(r)}] = \frac{\mathrm{Var}\left[n\bar{X} + m\bar{X}^{(r)}\right]}{(n+m)^2} = \frac{n^2\mathrm{Var}[\bar{X}] + m^2\mathrm{Var}[\bar{X}] + 2\mathrm{Cov}[n\bar{X}, m\bar{X}^{(r)}]}{(n+m)^2}, \tag{A.7}$$

where

$$\mathrm{Cov}[n\bar{X}_n, m\bar{X}_m^{(r)}] = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{i'=1}^{n}\mathrm{Cov}[X_i, X_{K_j}|K_j = i']P[K_j = i'] = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{i'=1}^{n}\frac{\sigma^2\delta_{i,i'}}{n} = m\sigma^2. \tag{A.8}$$

Thus, we find

$$\mathrm{Var}[\bar{X}_*^{(r)}] = \left[1 + \frac{m(n-1)}{(n+m)^2}\right]\frac{\sigma^2}{n}, \tag{A.9}$$

which is larger than $\mathrm{Var}[\bar{X}]$ but smaller than $\mathrm{Var}[\bar{X}^{(r)}]$.

A.2 Gaussian Generative Oversampling

Now consider points obtained by generative oversampling, wherein

$$X_j^{(g)} = \bar{X} + sZ_j, \tag{A.10}$$

where $X_j^{(g)}$ is the j^{th} point created by generative oversampling, s is the original sample standard deviation, and $Z_1, \ldots, Z_m$ are iid $N(0,1)$ and independent of $X_1, \ldots, X_n$ as well. For each resampled data point, the expected value is

$$E[X_j^{(g)}] = E[\bar{X}] + E[s]E[Z_j] = \mu + \sigma\sqrt{\frac{2}{n-1}}\frac{\Gamma(n/2)}{\Gamma((n-1)/2)}\cdot 0 = \mu, \tag{A.11}$$

while the variance is given by

$$\begin{aligned}\mathrm{Var}[X_j^{(g)}] &= \mathrm{Var}[\bar{X}] + \mathrm{Var}[sZ_j] + 2\mathrm{Cov}[\bar{X}, sZ_j] \\ &= \frac{\sigma^2}{n} + E[(sZ_j)^2] + 2E[(\bar{X}-\mu)sZ_j] \\ &= \frac{\sigma^2}{n} + E[s^2]E[Z_j^2] + 2E[(\bar{X}-\mu)s]E[Z_j] \\ &= \left(1+\frac{1}{n}\right)\sigma^2.\end{aligned} \tag{A.12}$$

Furthermore, the pair-wise covariance is

$$\begin{aligned}\mathrm{Cov}[X_j^{(g)}, X_{j'}^{(g)}] &= E[(\bar{X}+sZ_j-\mu)(\bar{X}+sZ_{j'}-\mu)] \\ &= E[(\bar{X}-\mu)^2] + E[(\bar{X}-\mu)s]\left(E[Z_j]+E[Z_{j'}]\right) + E[s^2]E[Z_jZ_{j'}] \\ &= \left(\delta_{j,j'}+\frac{1}{n}\right)\sigma^2\end{aligned} \tag{A.13}$$

The sample mean, $\bar{X}^{(g)}$, of the generatively resampled data is therefore unbiased, since

$$E[\bar{X}^{(g)}] = \frac{1}{m}\sum_{j=1}^{m} E[X_j^{(g)}] = \mu. \tag{A.14}$$

The variance of the sample mean may be computed as follows.

$$\mathrm{Var}[\bar{X}^{(g)}] = \frac{1}{m^2}\left[\sum_j \mathrm{Var}[X_j^{(g)}] + \sum_{j\neq j'}\mathrm{Cov}[X_j^{(g)}, X_{j'}^{(g)}]\right] = \left(1+\frac{n}{m}\right)\frac{\sigma^2}{n}. \tag{A.15}$$

Like the randomly resampled case, the variance of the sample mean is larger than that for the original sample.

Pooling the data leaves the sample mean, $\bar{X}_*^{(g)}$, unbiased. The variance of

this estimator is determined as follows. First, note that

$$\text{Var}[\bar{X}_*^{(g)}] = \frac{n^2\text{Var}[\bar{X}] + m^2\text{Var}[\bar{X}^{(g)}] + 2\text{Cov}[n\bar{X}, m\bar{X}^{(g)}]}{(n+m)^2} \tag{A.16}$$

where

$$\text{Cov}[n\bar{X}, m\bar{X}^{(g)}] = \sum_{i=1}^{n}\sum_{j=1}^{m}\text{Cov}[X_i, X_j^{(g)}] \tag{A.17}$$

and

$$\text{Cov}[X_i, X_j^{(g)}] = \text{Cov}[X_i, \bar{X} + sZ_j] = \text{Cov}[X_i, \bar{X}] + \text{Cov}[X_i, sZ_j]. \tag{A.18}$$

Now, note that

$$\text{Cov}[X_i, \bar{X}] = \frac{1}{n}\sum_{k=1}^{n}\text{Cov}[X_i, X_k] = \frac{1}{n}\sum_{k=1}^{n}\sigma^2\delta_{i,k} = \frac{\sigma^2}{n} \tag{A.19}$$

and

$$\text{Cov}[X_i, sZ_j] = E[(X_i - \mu)sZ_j] = E[(X_i - \mu)s]E[Z_j] = 0. \tag{A.20}$$

Thus, $\text{Cov}[X_i, X_j^{(g)}] = \sigma^2/n$, which implies $\text{Cov}[n\bar{X}_n, m\bar{X}_m^{(g)}] = m\sigma^2$. This, in turn, implies

$$\text{Var}[\bar{X}_*^{(g)}] = \left[1 + \frac{mn}{(n+m)^2}\right]\frac{\sigma^2}{n}. \tag{A.21}$$

As in the randomly resampled case, the variance of the sample mean is larger than that of the original sample.

A.3 Multinomial Random Oversampling

Let $X_1, \ldots, X_n$ be a random sample from a discrete distribution with values $1, \ldots, d$ and corresponding probabilities $\theta_1, \ldots, \theta_d$.[1] Let F_k denote the number of samples

[1] In the case of text mining, each X_i would correspond to a single word in the positive class and not a single document; in our analysis, the notation is much more straightforward if one looks

attaining the value $k \in \{1, \ldots, d\}$; i.e.,

$$F_k = \sum_{i=1}^{n} \delta_k(X_i), \tag{A.22}$$

where $\delta_k(x) = 1$ if $x = k$ and is zero otherwise. The random variables $F_1, \ldots, F_d$ then follow a multinomial distribution. The maximum likelihood estimator (MLE) $\hat{\theta}_k = F_k/n$ is unbiased for θ_k and has variance $\theta_k(1-\theta_k)/n$. With Laplace smoothing (a standard practice when using multinomials for problems like text mining), the estimator becomes

$$\tilde{\theta}_k = \frac{F_k + 1}{\sum_{k'=1}^{d}(F_{k'} + 1)} = \frac{n\hat{\theta}_k + 1}{n + d}, \tag{A.23}$$

where we have used the fact that $F_1 + \cdots + F_d = n$ almost surely. The smoothed estimator is biased, having $E[\tilde{\theta}_k] = (n\theta_k + 1)/(n + d)$, but has a smaller variance, $\mathrm{Var}[\tilde{\theta}] = n\theta_k(1 - \theta_k)/(n + d)^2$.

Now consider a randomly resampled data set $X_1^{(r)}, \ldots, X_m^{(r)}$, for which each $X_j^{(r)}$ is drawn uniformly and independently from the original sample. The number of resampled data points with value k will correspondingly be denoted $F_k^{(r)}$. For the estimator $\hat{\theta}_k^{(r)} = F_k^{(r)}/m$ we find

$$E[F_k^{(r)}] = \sum_{j=1}^{m} E[\delta_k(X_j^{(r)})] = \sum_{j=1}^{m}\sum_{i=1}^{n} P[X_j^{(r)} = k | X_j^{(r)} = x_i] P[X_j^{(r)} = x_i] = m\theta_k, \tag{A.24}$$

since $P[X_j^{(r)} = k | X_j^{(r)} = x_i] = \theta_k$ and $P[X_j^{(r)} = x_i] = 1/n$. For the variance, first note that

$$\mathrm{Var}[F_k^{(r)}] = \sum_{j=1}^{m} \mathrm{Var}[\delta_k(X_j^{(r)})] + \sum_{j \neq j'} \mathrm{Cov}[\delta_k(X_j^{(r)}), \delta_k(X_{j'}^{(r)})]. \tag{A.25}$$

at individual features instead of "blocks" of features that make up a datapoint (e.g., "blocks" of words making up a document); in addition, from the standpoint of parameter estimation, which document the word falls into does not matter.

Now,

$$\mathrm{Var}[\delta_k(X_j^{(r)})] = \sum_{i=1}^{n} \mathrm{Var}[\delta_k(X_j^{(r)})|X_j^{(r)} = x_i]\frac{1}{n} = \theta_k(1-\theta_k) \tag{A.26}$$

and

$$\begin{aligned}\mathrm{Cov}[\delta_k(X_j^{(r)}), \delta_k(X_{j'}^{(r)})] &= \sum_{i=1}^{n}\sum_{i'=1}^{n} \mathrm{Cov}[\delta_k(X_j^{(r)}), \delta_k(X_{j'}^{(r)})]|X_j^{(r)} = x_i, X_{j'}^{(r)} = x_{i'}]\frac{1}{n^2} \\ &= \frac{1}{n^2}\sum_{i} \mathrm{Cov}[\delta_k(X_i), \delta_k(X_{i'})] = \frac{\theta_k(1-\theta_k)}{n}.\end{aligned} \tag{A.27}$$

Combining these results, we find

$$\mathrm{Var}[\hat{\theta}_k^{(r)}] = \left(1 + \frac{n-1}{m}\right)\frac{\theta_k(1-\theta_k)}{n}. \tag{A.28}$$

For the pooled sample $X_1, \ldots, X_n, X_1^{(r)}, \ldots, X_m^{(r)}$, the MLE of θ is $(F_k + F_k^{(r)})/(n+m)$. Clearly, this estimator is unbiased. For the variance, note that

$$\mathrm{Var}\left[F_k + F_k^{(r)}\right] = \mathrm{Var}[F_k] + \mathrm{Var}[F_k^{(r)}] + 2\mathrm{Cov}[F_k, F_k^{(r)}]. \tag{A.29}$$

Now, the covariance is

$$\begin{aligned}\mathrm{Cov}[F_k, F_k^{(r)}] &= \sum_{i=1}^{n}\sum_{j=1}^{m} \mathrm{Cov}[\delta_k(X_i), \delta_k(X_j^{(r)})] \\ &= \sum_{i,j}\sum_{i'=1}^{n} \mathrm{Cov}[\delta_k(X_i), \delta_k(X_j^{(r)})|X_j^{(r)} = x_{i'}]\frac{1}{n} \\ &= \frac{1}{n}\sum_{i,i',j} \mathrm{Cov}[\delta_k(X_i), \delta_k(X_{i'})] = m\theta_k(1-\theta_k).\end{aligned} \tag{A.30}$$

After some simplification, we find that

$$\text{Var}\left[\frac{F_k + F_k^{(r)}}{n+m}\right] = \left[1 + \frac{m(n-1)}{(n+m)^2}\right]\frac{\theta_k(1-\theta_k)}{n}. \tag{A.31}$$

With Laplace smoothing, we have

$$E\left[\frac{F_k + F_k^{(r)} + 1}{n+m+d}\right] = \frac{(n+m)\theta_k + 1}{n+m+d} \tag{A.32}$$

and

$$\text{Var}\left[\frac{F_k + F_k^{(r)} + 1}{n+m+d}\right] = \left[1 + \frac{m(n-2d-1) - d(2n+d)}{(n+m+d)^2}\right]\frac{\theta_k(1-\theta_k)}{n}. \tag{A.33}$$

Thus, provided $m < d(2n+d)/(n-2d-1)$, the variance of the pooled, smoothed estimates will be smaller than that of the original sample, although the estimates themselves will be biased.

A.4 Multinomial Generative Oversampling

In generative oversampling, we use the probabilities estimated from $X_1, \ldots, X_n$ to generate a random sample $X_1^{(g)}, \ldots, X_m^{(g)}$ using the estimated parameters $\hat{\theta}_1, \ldots, \hat{\theta}_d$. The resampled data give unbiased estimates of the true parameters, since

$$E[F_k^{(g)}] = \sum_{j=1}^{m} E[\delta_k(X_j^{(g)})] = \sum_{j=1}^{m} P[X_j^{(g)} = k] \tag{A.34}$$

but

$$P[X_j^{(g)} = k] = \sum_{X_1,\ldots,X_n} P[X_j^{(g)} = k | X_1 = x_1, \ldots, X_n = x_n] P[X_1 = x_1, \ldots, X_n = x_n]$$
$$= \sum_{X_1,\ldots,X_n} \hat{\theta}_k P[X_1 = x_1, \ldots, X_n = x_n] = \theta_k. \tag{A.35}$$

For the variance, note that

$$\mathrm{Var}[F^{(g)}] = \sum_{j=1}^{m} \mathrm{Var}[\delta_k(X_j^{(g)})] + \sum_{j \neq j'} \mathrm{Cov}[\delta_k(X_j^{(g)}), \delta(X_{j'}^{(g)})]. \tag{A.36}$$

Now,

$$\mathrm{Var}[\delta_k(X_j^{(g)})] = E[\delta_k(X_j^{(g)})^2] - E[\delta_k(X_j^{(g)})]^2 = P[X_j^{(g)} = k] - P[X_j^{(g)} = k]^2 = \theta_k(1-\theta_k) \tag{A.37}$$

and

$$\mathrm{Cov}[\delta_k(X_j^{(g)}), \delta(X_{j'}^{(g)})] = P[X_j^{(g)} = k, X_{j'}^{(g)} = k] - \theta_k^2. \tag{A.38}$$

The joint probability, for $j \neq j'$, is

$$P[X_j^{(g)} = k, X_{j'}^{(g)} = k] = \sum_{X_1,\ldots,X_n} \hat{\theta}_k^2 P[X_1 = x_1, \ldots, X_n = x_n] = \frac{\theta_k(1-\theta_k)}{n} + \theta_k^2. \tag{A.39}$$

Thus,

$$\mathrm{Var}[f^{(g)}] = m\theta_k(1-\theta_k) + m(m-1)\frac{\theta_k(1-\theta_k)}{n}, \tag{A.40}$$

from which we readily deduce that

$$\mathrm{Var}[\hat{\theta}_k^{(g)}] = \left(1 + \frac{n-1}{m}\right) \frac{\theta_k(1-\theta_k)}{n}. \tag{A.41}$$

For the pooled sample $X_1, \ldots, X_n, X_1^{(g)}, \ldots, X_m^{(g)}$ we again have an unbiased estimator. The variance is given by

$$\mathrm{Var}[F_k + F_k^{(g)}] = \mathrm{Var}[F_k] + \mathrm{Var}[F_k^{(g)}] + 2\mathrm{Cov}[F_k, F_k^{(g)}], \tag{A.42}$$

where

$$\begin{aligned}\mathrm{Cov}[F_k, F_k^{(g)}] &= \sum_{i=1}^{n}\sum_{j=1}^{m}\mathrm{Cov}[\delta_k(X_i), \delta_k(X_j^{(g)})] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{m}\left\{P[X_i = k, X_j^{(g)} = k] - P[X_i = k]P[X_j^{(g)} = k]\right\}.\end{aligned} \tag{A.43}$$

Now, the joint probability is found to be

$$\begin{aligned}P[X_i = k, X_j^{(g)} = k] &= \sum_{X_1,\ldots,X_n} P[X_i = k, X_j^{(g)} = k | X_1 = x_1, \ldots, X_n = x_n]P[X_1 = x_1, \ldots, X_n = x_n] \\ &= \sum_{X_1,\ldots,X_n} \delta_k(X_i)\hat{\theta}_k P[X_1 = x_1, \ldots, X_n = x_n].\end{aligned} \tag{A.44}$$

But

$$\sum_{i=1}^{n} P[X_i = k, X_j^{(g)} = k] = \sum_{X_1,\ldots,X_n} n\hat{\theta}_k\hat{\theta}_k P[X_1 = x_1, \ldots, X_n = x_n] = \theta_k(1-\theta_k)+n\theta_k^2, \tag{A.45}$$

so

$$\mathrm{Cov}[F_k, F_k^{(g)}] = m\theta_k(1-\theta_k). \tag{A.46}$$

Combining these results, we find

$$\mathrm{Var}[F_k + F_k^{(g)}] = n\theta_k(1-\theta_k) + m^2\left(1 + \frac{n-1}{m}\right)\frac{\theta_k(1-\theta_k)}{n} + 2m\theta_k(1-\theta_k), \tag{A.47}$$

and, thus,

$$\text{Var}\left[\frac{F_k + F_k^{(g)}}{n+m}\right] = \left[1 + \frac{m(n-1)}{(n+m)^2}\right] \frac{\theta_k(1-\theta_k)}{n}. \tag{A.48}$$

These results are identical to those of the random oversampling case.

Appendix B

Estimated Mean by Uncertainty Sampling

Let $q_t \in \mathcal{R}$ be the queried point on the tth iteration of active learning. Then $P(y_+|q_t) = P(y_-|q_t) = \frac{1}{2}$ because we pick a point with maximum uncertainty. Depending on the class label of the queried point, $\mu_{+,t}$ and $\mu_{-,t}$ have $\frac{1}{2}$ chance of updates:

$$E[\hat{\mu}_{\cdot,t}] = \hat{\mu}_{\cdot,t-1} + \frac{1}{2t}(q_t - \hat{\mu}_{\cdot,t-1}) \quad .$$

Thus, $E[\hat{\mu}_{+,t}]$ and $E[\hat{\mu}_{-,t}]$ move towards q_t with step-size $1/2t$. Since we are running query-based active learning with uncertainty sampling, we can choose q_t with maximum uncertainty condition: $P_{t-1}(y_+|q_t) = P_{t-1}(y_-|q_t)$, where $P_{t-1}(y|q_t)$ is the posterior probability estimated after $t-1$ steps. Since we assumed equal variances

and priors for both classes, we can easily get

$$
\begin{aligned}
E[q_t] &= \frac{1}{2}(E[\hat{\mu}_{+,t-1}] + E[\hat{\mu}_{-,t-1}]) \quad \text{and} \\
E[q_{t+1}] &= \frac{1}{2}\left(\hat{\mu}_{+,t-1} + \frac{1}{2t}(q_t - \hat{\mu}_{+,t-1}) + \hat{\mu}_{-,t-1} + \frac{1}{2t}(q_t - \hat{\mu}_{-,t-1})\right) \\
&= \frac{1}{2}\hat{\mu}_{+,t-1} + \frac{1}{2}\hat{\mu}_{-,t-1} + \frac{1}{2t}q_t - \frac{1}{2t}\frac{1}{2}(\hat{\mu}_{+,t-1} + \hat{\mu}_{-,t-1}) \\
&= q_t + \frac{1}{2t}q_t - \frac{1}{2t}q_t = q_t \ .
\end{aligned}
$$

Thus, q_t is constant for all iterations t. Since $\mathrm{E}[\hat{\mu}_{\cdot,t}]$ converges to q_t, and q_t is a constant, $\hat{\mu}_{+,t}$ gets farther and farther away from $\mu_{\cdot}$ on each iteration of active learning. Moreover, one can similarly show that q_t falls precisely on the decision boundary, which is also constant for all t. This decision boundary is at $q_t = q_0$, which, in expectation, can be shown to be the true decision boundary that minimizes Bayesian error rate (i.e., halfway between the true means μ_+ and μ_-).

Appendix C

Code

The latest instructions for obtaining code used to run experiments described in this dissertation can be obtained at: http://www.ideal.ece.utexas.edu/~aliu/code/index.html

Bibliography

[AN07] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[BCNM06] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 535–541, 2006.

[BDL09] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *ICML '09: Proceedings of the 26th international conference on Machine learning*, 2009.

[CBHK02] Nitesh Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling TEchnique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[CCG07] Yangchi Chen, Melba M. Crawford, and Joydeep Ghosh. Knowledge based stacking of hyperspectral data for land cover classification. *IEEE Symposium on Computational Intelligence and Data Mining '07*, pages 316–322, 2007.

[CNM04] Rich Caruana and Alexandru Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *KDD '04: Proceedings of the tenth ACM SIGKDD international*

conference on Knowledge discovery and data mining, pages 69–78, New York, NY, USA, 2004. ACM Press.

[CS98] Philip K. Chan and Salvatore J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. *Knowledge Discovery and Data Mining*, pages 164–168, 1998.

[DFG01] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In V. Kumar R. Grossman, C. Kamath and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001. Invited book chapter.

[DH06] Chris Drummond and Robert C. Holte. Cost curves: An improved method for visualizing classifier performance. *Machine Learning*, 65(1):95–130, 2006.

[DH08] Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on machine learning*, pages 208–215, 2008.

[DHM08] Sanjoy Dasgupta, Daniel Hsu, and Claire Monteleoni. A general agnostic active learning algorithm. In *Advances in Neural Information Processing Systems 20*, pages 353–360, 2008.

[DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, 2001.

[Dom99] Pedro Domingos. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, 1999.

[EJJ04] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20, number 1, 2004.

[Elk01] Charles Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

[FDG05] Dominique Feillet, Pierre Dejax, and Michel Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39:188–205, 2005.

[Fla03] Peter A. Flach. The geometry of ROC space: understanding machine learning metrics through ROC isometrics. In *ICML '03: Proceedings of the Twentieth International Conference on Machine Learning*, pages 194–201, January 2003.

[GLS98] Michel Gendreau, Gilbert Laporte, and Frederic Semet. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, 32:263–273, 1998.

[Han09] David Hand. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning Journal*, 77(1):103–123, 2009.

[HBG+98] Eui-Hong Han, Daniel Boley, Maria Gini, Robert Gross, Kyle Hastings, George Karypis, Vipin Kumar, Bamshad Mobasher, and Jerome Moore. WebACE: A web agent for document categorization and exploration. *Proceedings of the Second International Conference on Autonomous Agents*, pages 408–415, 1998.

[HBLH94] William Hersh, Chris Buckley, T. J. Leone, and David Hickam.

Ohsumed: An interactive retrieval evaluation and new large test collection for research. *Proceedings of ACM SIGIR*, pages 192–201, 1994.

[HCCG05] Jisoo Ham, Yangchi Chen, Melba M. Crawford, and Joydeep Ghosh. Investigation of the random forest framework for classification of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):492–501, 2005.

[HKN07] Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 935–942, 2007.

[HSRC08] Robbie Haertel, Kevin D. Seppi, Eric K. Ringger, and James L. Carroll. Return on investment for active learning. *NIPS Workshop on Cost-sensitive Learning*, 2008.

[Jap06] Nathalie Japkowicz. Why question machine learning evaluation methods? *AAAI-06 Evaluation Methods for Machine Learning Workshop*, pages 6–11, 2006.

[JL98] Luis O. Jimenez and David A. Landgrebe. Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 28(1):39–54, Feb 1998.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference*

on Machine Learning, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[Kar02] George Karypis. CLUTO - a clustering toolkit. *University of Minnesota technical report 02-017*, 2002.

[KG07] Andreas Krause and Carlos Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 449–456, New York, NY, USA, 2007. ACM.

[KGC01] Shailesh Kumar, Joydeep Ghosh, and Melba M. Crawford. Best-bases feature extraction algorithms for classification of hyperspectral data. *IEEE Trans. on Geosci. and Remote Sens.*, 39(7):1368–1379, 2001.

[KHM98] Miroslav Kubat, Robert C. Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2-3):195–215, 1998.

[Lan02] David A. Landgrebe. Hyperspectral image data analysis. *Signal Processing Magazine, IEEE*, 19(1):17–28, Jan 2002.

[LC94] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1994.

[LL98] Charles X. Ling and Chenghui Li. Data mining for direct marketing: Problems and solutions. *Knowledge Discovery and Data Mining*, pages 73–79, 1998.

[LLKS85] E. L. Laweler, Jan Karel Lenstra, A. H. G. Rinnooy Kan, and D. B.

Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, 1985.

[Mal03] Marcus Maloof. Learning when data sets are imbalanced and when costs are unequal and unknown. *ICML-2003 Workshop on Learning from Imbalanced Data Sets II*, 2003.

[Mar05] Dragos D. Margineantu. Active cost-sensitive learning. In *the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[MB04] Farid Melgani and Lorenzo Bruzzone. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 42(8):1778–1790, Aug. 2004.

[MBJ99] Katharina Morik, Peter Brockhausen, and Thorsten Joachims. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999.

[MM04] Prem Melville and Raymond J. Mooney. Diverse ensembles for active learning. *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, pages 584–591, 2004.

[MN98] Andrew McCallum and Kamal Nigam. A comparison of event models for Naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[Mor02] Joseph T. Morgan. *Adaptive hierarchical classifier with limited training data*. PhD thesis, Univ. of Texas at Austin, 2002.

[MZW05] Kate McCarthy, Bibi Zabar, and Gary Weiss. Does cost-sensitive learning beat sampling for classifying rare classes? *UBDM '05: Proceedings*

of the 1st international workshop on Utility-based data mining, pages 69–77, 2005.

[PAL04] Clifton Phua, Damminda Alahakoon, and Vincent Lee. Minority report in fraud detection: Classification of skewed data. *SIGKDD Explor. Newsl.*, 6(1):50–59, 2004.

[PFK98] Foster Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453, 1998.

[RGC08] Suju Rajan, Joydeep Ghosh, and Melba M. Crawford. An active learning approach to hyperspectral data classification. *IEEE Transactions on Geoscience and Remote Sensing*, 46(4):1231–1242, 2008.

[RM01] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning*, pages 441–448. Morgan Kaufmann Publishers Inc., 2001.

[SC00] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA, 2000.

[SCF08] Burr Settles, Mark Craven, and Lewis Friedland. Active learning with real annotation costs. *NIPS Workshop on Cost-sensitive Learning*, 2008.

[Set09] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[SJS06] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, F-Score and ROC: A family of discriminant measures for per-

formance evaluation. In *Australian Conference on Artificial Intelligence*, pages 1015–1021, 2006.

[SOS92] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 287–294, Pittsburgh, PA, USA, 1992. ACM Press.

[STP04] Maytal Saar-Tsechansky and Foster Provost. Active sampling for class probability estimation and ranking. *Mach. Learn.*, 54(2):153–178, 2004.

[STP07] Maytal Saar-Tsechansky and Foster Provost. Decision-centric active learning of binary-outcome models. *Information Systems Research*, 18(1):1–19, 2007.

[TK02] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, 2002.

[Tur00] Peter D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.

[VO00] Ricardo Vilalta and Daniel Oblinger. A quantification of distance-bias between evaluation metrics in classification. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1087–1094. Morgan Kaufmann, San Francisco, CA, 2000.

[VR05] Sofia Visa and Anca Ralescu. Issues in mining imbalanced data sets - a review paper. In *Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference*, pages 67–73, 2005.

[WMZ07] Gary Weiss, Kate McCarthy, and Bibi Zabar. Cost-sensitive learning vs. sampling: Which is best for handling class imbalance? *DMIN '07: International Conference on Data Mining*, 2007.

[WP03] G. Weiss and F. Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–324, 2003.

[Yan99] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):69–90, 1999.

[Yar95] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196, 1995.

[ZG03] Shi Zhong and Joydeep Ghosh. A comparative study of generative models for document clustering. *SDM Workshop on Clustering High Dimensional Data and Its Applications*, 2003.

[Zhu05] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[ZLA03] Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. *Proc. 3rd ICDM*, pages 435–442, 2003.

[ZWS04] Zhaohui Zheng, Xiaoyun Wu, and Rohini K. Srihari. Feature selection for text categorization on imbalanced data. *SIGKDD Explorations*, 6(1):80–89, 2004.

Vita

Alexander Liu received his BS and MS degrees in Electrical and Computer Engineering from the University of Texas at Austin in 2001 and 2004, respectively. He has been supported by a number of fellowships, including the Texas Excellence Award for Scholarship and Leadership, a Microelectronics and Computer Development Fellowship, and the Thrust Fellowship. His research interests include data mining and machine learning, particularly topics in utility-based data mining such as cost-sensitive learning and active learning.

Permanent Address: 14206 Redbud Valley Trail
Houston, TX 77062

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society.